KAPITOLA 11

Techniky antivirové obrany

"Ale kdo bude hlídat hlídače?"

– Juvenal

Tato kapitola je kolekcí technik, které se používají v antivirovém software pro ochranu uživatelů před počítačovými viry. Postupně budou vysvětleny techniky antivirových skenerů, které se vyvinuly (společně s počítačovými viry) během posledních patnáct let. Během dlouhého vývoje antivirového software se tyto techniky vyladily a dnes jsou veřejně známé a používané. Ačkoliv se časem určitě objeví nové postupy, metody uvedené v této kapitole, se jeví pro dohlednou budoucnost jako dostatečné.

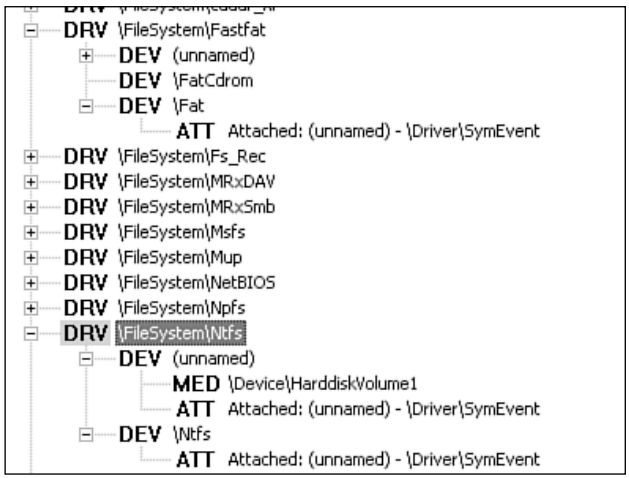
Pro lepší přehlednost bude detekce počítačových virů rozdělena na tyto tři části:

- Detekce založená na prostém vyhledávání vzorků.
- Přesná identifikace.
- Detekce zakódovaných, polymorfních a metamorfních virů[1].

Také vám ukáži použití generických a heuristických metod[2], které mohou detekovat celé třídy počítačových virů a nikoliv pouze jejich varianty. Tato kapitola vás také seznámí s technikami léčení (včetně generických a heuristických metod), které se používají pro opravu infikovaných souborů. Současný antivirový software pro heuristickou analýzu[3] používá sofistikovanou emulaci kódu (tzv. virtuální stroj) pro komplexní detekci virů. Jedná se o klíčovou komponentu antivirového software, která pomohla udržet antivirové skenery dlouho při životě.

Existují dva základní druhy skenerů: on-demand a on-access. On-demand skenování se spouští pouze na příkaz uživatele a dá se také spouštět během startu operačního systému. On-access skenery jsou naopak neustále rezidentní v paměti a nahrávají se jako jednoduché aplikace, které se zavěsí na přístup k diskům a souborům nebo jsou implementovány jako ovladače zařízení, které se připojují na souborové systémy[4]. Například na systémech Windows NT/2000/XP/2003 jsou on-access skenery typicky implementovány jako ovladače filtrů souborového systému (file-system filter drivers), které jsou připojeny na použité souborové systémy, jako například FAT, NTFS atd.

Obrázek 11.1 znázorňuje načtený ovladač filtru souborového systému připojený na sadu souborových systémů s použitím nástroje z OSR.



Obrázek 11.1 *Ovladač filtru souborového systému připojený k ovladačům souborového systému.*

On-access skenery obvykle skenují soubory při jejich otevírání, vytváření nebo zavírání. Takto se dá zabránit tomu, aby se na systému spustil známý virus. Zajímavý problém vzniká u síťových infektorů, jakým je třeba virus W32/Funlove. Funlove totiž infikuje soubory na sdílených síťových discích, takže infekce na vzdáleném systému bude detekována pouze v případě, že je soubor zapsán na disk. To znamená, že v některých případech mohou mít on-access skenery problémy při zastavování činnosti virů.

Poznámka

Toto riziko se dá omezit skenováním diskové cache ještě před tím, než je soubor zapsán na disk. Mohou se pochopitelně použít i jiné metody obrany, jako třeba monitory podezřelého chování nebo software zabraňující proniknutí do sítě.

Tato kapitola se také zaměřuje na techniky, které dovedou infekcím zabránit a vyléčit je – jak v souborech, tak i v oblastech souborových systémů. Také se podíváme na obecně použitelná řešení, která zahrnují následující:

- On-demand kontrolory integrity dat.
- On-access kontrolory integrity dat (tzv. integrity shells).
- Monitory podezřelého chování (tzv. behavior blockers).
- Kontrolu přístupu.
- Očkování.

11.1 Skenery první generace

Většina počítačových knih rozebírá detekci virů na celkem jednoduché úrovni. Dokonce i novější knihy popisují antivirové skenery jako jednoduché programy, které prohledávají sekvence bajtů extrahované z počítačového viru umístěného v souboru nebo v paměti. Jedná se skutečně o jednu z nejpopulárnějších metod pro detekci počítačových virů, která je navíc v jistých ohledech efektivní. Současný antivirový software používá k detekci složitých virů mnohem zajímavější techniky, které skenery první generace nezvládají. Následující části textu rozebírají příklady metod detekce a identifikace, které se dají použít k detekci počítačových virů.

Poznámka

Ne všechny techniky se dají použít k detekci všech počítačových virů, na druhou stranu se to ani neočekává. Stačí mít arzenál vlastní detekčních technik, z nichž jedna dobře poslouží k detekci nebo dezinfekci příslušného viru. Tento fakt je často opomíjen bezpečnostními specialisty a výzkumníky, kteří jsou schopni se hádat o efektivitě techniky v případě, že se nedá použit na detekci všech infekcí.

11.1.1 Skenování řetězců

Skenování řetězce je tím nejjednodušším způsobem, jakým lze detekovat počítačové viry. Je použito extrahovaných sekvencí bajtů (řetězců), které jsou typické pro konkrétní virus, a které se nenacházejí v čistých programech. Tyto sekvence se ukládají v databázích, které antivirové skenery systematicky používají při prohledávání v předdefinovaných oblastech souborů a v systémových oblastech, aby tak v omezeném čase, který mají vyhrazen na skenování, detekovaly počítačové viry. A skutečně – jedna z nejdůležitějších výzev, kterou před sebou enginy antivirových skenerů mají, je efektivní využití omezeného času k provedení testu (obvykle to nebývá více než pár sekund na jeden soubor).

Podívejte se na část kódu uvedeného na obrázku 11.2, který byl získán pomocí nástroje IDA (Interactive DisAssembler) z jedné varianty boot viru Stoned.

```
seg000:7C40 BE 04 00                    mov     si, 4           ; Try it 4 times
seg000:7C40                                                     ;
seg000:7C43
seg000:7C43                 next:                               ; CODE XREF: sub_7C3A+27↓j
seg000:7C43 B8 01 02                    mov     ax, 201h        ; read one sector
seg000:7C46 0E                          push    cs
seg000:7C47 07                          pop     es
seg000:7C48                             assume es:seg000
seg000:7C48 BB 00 02                    mov     bx, 200h        ; to here
seg000:7C4B 33 C9                       xor     cx, cx
seg000:7C4D 8B D1                       mov     dx, cx
seg000:7C4F 41                          inc     cx
seg000:7C50 9C                          pushf
seg000:7C51 2E FF 1E 09 00              call    dword ptr cs:9  ; int 13
seg000:7C56 73 0E                       jnb     short fine
seg000:7C58 33 C0                       xor     ax, ax
seg000:7C5A 9C                          pushf
seg000:7C5B 2E FF 1E 09 00              call    dword ptr cs:9  ; int 13
seg000:7C60 4E                          dec     si
seg000:7C61 75 E0                       jnz     short next
seg000:7C63 EB 35                       jmp     short giveup
```

Obrázek 11.2 *Část kódu viru Stoned, který byl načtený do programu IDA.*

Tato část kódu čtyřikrát přečte boot sektor diskety a před každým pokusem zresetuje disk.

> **Poznámka**
>
> Virus potřebuje volat původní obsluhu INT 13, protože současně monitoruje stejné přerušení, aby mohl infikovat diskety kdykoliv, když se k nim přistoupí. Proto virus volá CS:[09] přímo v datové oblasti na počátku virového kódu, na 0:7C09, kde se nachází uložená adresa původní diskové obsluhy. Na začátku virového kódu je pár bajtů dat, zbytek těla zůstává konstantní.

Jedná se o typickou sekvenci kódu viru. Čtyři pokusy čtení prvního sektoru jsou nezbytné kvůli starším disketovým mechanikám, které byly příliš pomalé. Virus používá dvojici instrukcí PUSH CS, POP ES, aby nastavil diskový zásobník (buffer) na segment viru.

Styl kódu vypadá jako pokus o optimalizaci nastavení obsahu registrů CX a DX, které slouží jako parametry volání přerušení diskové obsluhy.

Následujících 16 bajtů, což je část kódu vyextrahovaná z viru Stoned, může sloužit jako vyhledávací řetězec pro tento virus. Byl publikován v magazínu Virus Bulletin.

```
0400 B801 020E 07BB 0002 33C9 8BD1 419C
```

Těchto šestnáct jedinečných bajtů je dostatečně dlouhým řetězcem pro bezpečnou detekci 16bitového škodlivého kódu bez rizika falešných poplachů. Není tedy překvapením, že se pro detekci různých DOSových a boot virů používaly sekvence, které byly dlouhé právě těch šestnáct bajtů. Sekvence bývaly také publikovány v různých magazínech zaměřených na antivirový výzkum (například Virus Bulletin). Pro bezpečnou detekci 32bitových virů jsou obvykle zapotřebí delší vyhledávací řetězce, zejména v případě, kdy je kód napsaný v nějakém vysokoúrovňovém jazyce.

Předchozí sekvence kódu se může objevit také v jiných variantách viru Stoned. Tímto řetězcem se mohou detekovat nejenom varianty viru A, B a C, dají se jím také detekovat příbuzné viry, které už spadají do jiné rodiny. Na jednu stranu je to výhoda, protože skener dokáže jedním řetězcem detekovat více virů, na druhou stranu se může stát, že tímto řetězcem bude detekován úplně jiný virus, který bude nesprávně označen jako virus Stoned. Uživatel si pak může chybně myslet, že tento virus je relativně neškodný, ovšem špatná identifikace může způsobit mnohem více škod.

Špatná identifikace viru může vést k závažným komplikacím. Ty mohou nastat třeba v okamžiku, kdy se antivirový skener pokusí napadený objekt vyléčit. Protože postup odstranění dvou odlišných rodin virů nebo i dvou variant stejného viru je obvykle odlišný, mohou snadno nastat problémy.

Pro vysvětlení – některé varianty viru Stoned například ukládají původní MBR na sektor 7, přičemž jiné varianty používají sektor 2. Pokud antivirový program neidentifikuje virus správně (alespoň ve své dezinfekční rutině), výsledkem bude to, že po dezinfekci nebude možné se nabootovat do systému.

Existují některé techniky, které umí omezit vznik těchto komplikací. Některé jednoduché dezinfektory používají v opravném kódu různé záložky k tomu, aby se ujistily, že dezinfekční kód je opravdu určen konkrétní variantě viru.

11.1.2 Zástupné znaky

Jednoduché skenery často používají tzv. zástupné znaky (wildcards), které obvykle umožňují přeskočit bajty nebo rozsah bajtů. Některé skenery navíc podporují regulární výrazy. Viz tento řetězec:

```
0400 B801 020E 07BB ??02 %3 33C9 8BD1 419C
```

Tento řetězec se bude interpretovat následovně:

1. Porovnej na 04 a pokud se shoduje, pokračuj.
2. Porovnej na 00 a pokud se shoduje, pokračuj.
3. Porovnej na B8 a pokud se shoduje, pokračuj.
4. Porovnej na 01 a pokud se shoduje, pokračuj.
5. Porovnej na 02 a pokud se shoduje, pokračuj.
6. Porovnej na 0E a pokud se shoduje, pokračuj.
7. Porovnej na 07 a pokud se shoduje, pokračuj.
8. Porovnej na BB a pokud se shoduje, pokračuj.
9. Ignoruj tento bajt.

10. Porovnej na 02 a pokud se shoduje, pokračuj.

11. Porovnej na 33 na následujících třech pozicích a pokud se shoduje, pokračuj.

12. Porovnej na C9 a pokud se shoduje, pokračuj.

13. Porovnej na 8B a pokud se shoduje, pokračuj.

14. Porovnej na D1 a pokud se shoduje, pokračuj.

15. Porovnej na 41 a pokud se shoduje, pokračuj.

16. Porovnej na 9C a pokud se shoduje, nahlas infekci.

Zástupné znaky se často používají pro půlbajty, které umožňují přesnější porovnání skupin instrukcí. Některé z prvních zakódovaných virů (a dokonce i polymorfní viry) bylo možné jednoduše detekovat řetězci na bázi zástupných znaků.

Použití samotného Boyer-Moorova algoritmu[5] není ovšem dostatečně efektivní pro skenery, které jsou založeny na řetězcích. Tento algoritmus byl vyvinut pro rychlé prohledávání řetězců a využívá porovnávání řetězců pozpátku. Uvažme následující dvě slova se stejnou délkou:

```
CONVENED a CONVENER
```

Pokud jsou dva řetězce porovnávány od začátku, je pro nalezení rozdílu potřeba celkem sedm porovnání. Pokud se začne od konce řetězce, rozdíl odhalí už první porovnání, což významně redukuje celkový počet porovnání, které je potřeba učinit.

> **Poznámka**
>
> Boyer-Moorův algoritmus nefunguje příliš dobře v systémech IDS (Intrusion Detection Systems, systémy pro detekci průniku), protože tento způsob porovnávání může způsobit přetečení v paketu.

Podobného úspěchu je možné dosáhnout na základě použití technik záložek, které jsou vysvětleny později. Navíc s použitím různých filtrovacích[6] a hašovacích algoritmů může být rychlost virtuálně nezávislá na celkovém počtu řetězců, které se musí porovnávat.

11.1.3 Neshody

Neshody v řetězcích byly vymyšleny pro IBM Antivirus. Ty umožňují, aby N bajtů v řetězci mohlo být libovolné, bez ohledu na jejich umístění v řetězci. Například řetězec 01 02 03 04 05 07 08 09 s hodnotou neshody 2 se bude shodovat ve všech následujících vzorcích (viz obrázek 11.3).

```
01 02 AA 04 05 06 BB 08 09 0A 0B 0C 0D 0E 0F 10
01 02 03 CC DD 06 07 08 09 0A 0B 0C 0D 0E 0F 10
01 EE 03 04 05 06 07 FF 09 0A 0B 0C 0D 0E 0F 10
```

Obrázek 11.3 *Sada řetězců, které se liší ve dvou neshodách.*

Neshody jsou užitečné zejména při vytváření generických řešení pro detekce rodin počítačových virů, nevýhodou algoritmu je jeho pomalost během skenování.

11.1.4 Generická detekce

Generická detekce skenuje několik nebo všechny varianty rodiny počítačových virů s použitím prostého řetězce. Jakmile je nalezena více než jedna varianta počítačového viru, sada variant je analyzována v obvyklé oblasti kódu, přičemž se vybere jednoduchý vyhledávací řetězec, který je přítomný v co nejvíce variantách. Generický řetězec obvykle obsahuje jak zástupné znaky, tak i neshody.

11.1.5 Hašování

Hašování je běžný termín popisující techniky, které urychlují vyhledávací algoritmy. Hašování se dá udělat pro první bajt nebo 16bitová a 32bitová slova vyhledávacího řetězce, což umožňuje, aby další bajty obsahovaly zástupné znaky. Antiviroví výzkumníci mohou hašování ještě vylepšit tak, že zvolí nějaký začínající bajt, který bude řetězec obsahovat. Je dobré vyhnout se prvním bajtům, které jsou běžné v normálních souborech, jako jsou třeba nuly. Výzkumník může vybrat řetězce, které obvykle začínají stejnými společnými bajty, což redukuje počet nutných porovnání.

Aby bylo vyhledávání co nejrychlejší, některé skenery vůbec nepodporují zástupné znaky. Například australský antivirus VET používá vynález Rogera Riordana[7], který je založen na použití 16bajtových vyhledávacích řetězců (bez zástupných znaků) založených na 64 kB velké hašovací tabulce a 8bitového posuvného registru. Tento algoritmus pak použije každé 16bitové slovo řetězce jako index do hašovací tabulky.

Velmi silné hašování bylo vyvinuto Fransem Veldmanem v TBSCAN. Tento algoritmus používá v řetězcích zástupné znaky, přičemž má dvě hašovací tabulky a odpovídající spojitý seznam řetězců. První hašovací tabulka obsahuje indexové bity do druhé tabulky. Algoritmus je založen na použití čtyř konstantních 16bitových nebo 32bitových slov vyhledávacího řetězce, který v sobě neobsahuje žádné zástupné znaky.

11.1.6 Záložky

Záložky (nebo také kontrolní bajty) slouží jako jednoduchý způsob pro zajištění přesnější detekce a dezinfekce. Obvykle se v bajtech vypočítá vzdálenost mezi začátkem těla viru (neboli nultým bajtem těla) a detekčním řetězcem a uloží se odděleně v detekčních záznamech.

Dobré záložky jsou specifické pro dezinfekci viru. Například v případě boot virů může dát někdo přednost vybrání sady záložek, které ukazují na reference v uložených boot sektorech. Když použijeme předchozí příklad viru Stoned, vzdálenost mezi začátkem těla viru a řetězcem je 0x41 (65) bajtů. Nyní se podívejte na část kódu na obrázku 11.4. Kód přečte uložený boot sektor v závislosti na příznaku. V případě pevného disku je načten uložený boot sektor a spustí se z hlavy 0, stopy 0 a sektoru 7 z jednotky C:. V případě disket se načte sektor kořenového adresáře z hlavy 0, stopy 3 a sektoru 1 z jednotky A:.

```
seg000:7CE9 33 C0                   xor     ax, ax
seg000:7CEB 8E C0                   mov     es, ax
seg000:7CED                         assume es:seg000
seg000:7CED B8 01 02                mov     ax, 201h
seg000:7CF0 BB 00 7C                mov     bx, 7C00h
seg000:7CF3 2E 80 3E 08 00 00       cmp     byte ptr cs:8, 0 ; which drive?
seg000:7CF9 74 0B                   jz      short diskette
seg000:7CFB B9 07 00                mov     cx, 7           ; hard disk
seg000:7CFE BA 80 00                mov     dx, 80h ; 'Ç'
seg000:7D01 CD 13                   int     13h             ; DISK - READ SECTORS INTO MEMORY
seg000:7D01                                                 ; AL = number of sectors to read
seg000:7D01                                                 ; CH = track, CL = sector
seg000:7D01                                                 ; DH = head, DL = drive,
seg000:7D01                                                 ; ES:BX -> buffer to fill
seg000:7D01                                                 ; Return: CF set on error,
seg000:7D01                                                 ; AH = status, AL = number of sectors
seg000:7D03 EB 49                   jmp     short exit
seg000:7D05                 ; ---------------------------------------------------------------------------
seg000:7D05 90                      nop
seg000:7D06
seg000:7D06                 diskette:                       ; CODE XREF: seg000:7CF9↑j
seg000:7D06 B9 03 00                mov     cx, 3
seg000:7D09 BA 00 01                mov     dx, 100h
seg000:7D0C CD 13                   int     13h
```

Obrázek 11.4 *Další část kódu viru Stoned načtená do IDA.*

Jako postačující sada záložek mohou sloužit následující bajty:

- První záložku můžeme vybrat z offsetu 0xFC (252) těla viru, kde se nachází bajt 0x07.
- Druhou záložku můžeme vybrat z offsetu 0x107 (263) těla viru, kde se nachází bajt 0x03.

Tyto bajty můžete nalézt na offsetech 0x7CFC a 0x7D07 v předešlém disassemblovaném výpisu. Pamatujte si, že se tělo viru načítá na offset 0x7C00.

Poznámka

V případě souborových virů je dobré vybrat takové záložky, které ukazují na offset, kde je uložená hlavička původního hostitelského programu. Další dobrou záložkou může být také velikost těla viru.

Nekorektní oprava zavirovaného souboru se dá bezpečně omezit kombinací vyhledávacího řetězce a záložek. V praxi je často bezpečné opravit virus na základě těchto informací, nicméně další přesná a téměř přesná identifikace viru může detekci ještě více vylepšit.

11.1.7 Skenování začátku a konce

Skenování začátku a konce se často používá pro zrychlení detekce virů, a to skenováním pouze začátku a konce souboru, namísto skenování celého souboru. Proskenují se například první a poslední 2, 4 nebo 8 kB souboru na každé možné pozici. Jedná se o trochu lepší algoritmus než ty, které používaly první implementace antivirových skenerů, které pracovaly velmi podobně jako program GREP (který prohledává celý soubor na výskyt shodného řetězce). Jak se moderní procesory staly rychlejšími, začala být rychlost skenování záviset na rychlosti I/O operací. Pro optimalizaci rychlosti se vývojáři antivirových programů zaměřili na metody, které dokážou zredukovat počet čtení z disku. Protože se většina prvních počítačových virů připojovala na začátek nebo na konec hostitelských programů, stalo se skenování začátku a konce souboru docela populární technikou.

11.1.8 Skenování vstupních a fixních bodů

Skenování vstupních a fixních bodů učinilo antivirové skenery ještě rychlejšími. Takové skenery využívají vstupní body objektů, jako třeba ty, které jsou přístupné přes hlavičky spustitelných souborů. V binárních spustitelných souborech bez struktur, jako třeba DOSové COM soubory, takové skenery následující různé instrukce, které předávají řízení (jako třeba instrukce skoku a volání) a skenují místa, kam tyto instrukce ukazují.

Protože jsou tato místa častým cílem počítačových virů, mají takové skenery velkou výhodu. Jiné skenovací metody, jako třeba již zmiňované skenování začátku a konce musí porovnávat řetězce (nebo haše řetězců) s každou možnou pozicí ve skenovací oblasti, ale skenery vstupních bodů mají obvykle jedinou pozici pro skenování – vlastní vstupní bod.

Uvažte 1 kB dlouhý buffer nazvaný jako B. Počet pozic, na kterých se má porovnávat řetězec, je 1024 minus S, kde S je velikost nejkratšího řetězce na porovnání. I když je hašovací algoritmus skeneru tak efektivní, že skener potřebuje provést kompletní vyhledávání řetězce pouze v 1% času, počet výpočtů se může rychle vyšplhat nahoru (v závislosti na počtu řetězců). Například s 1000 řetězců bude skener potřebovat učinit 10 kompletních porovnání na každé možné pozici. Bude tedy potřeba vykonat minimálně (1,024 – S)x10 porovnání. Násobek 1024 – S může být snížen skenováním fixního bodu s jediným potřebným porovnáním ve vstupním bodě. Jedná se tedy o velmi významný rozdíl.

Jestliže vstupní bod nemá dostatečně dobré řetězce, skenování ve fixním bodě může pomoct. Skenování ve fixním bodě porovnává pozici s každým řetězcem, takže je možné nastavit počátek M (například hlavní vstupní bod programu) a pak porovnávat každý řetězec (nebo haš) na pozici M + X bajtů od tohoto fixního bodu. Takto se opět redukuje počet nutných výpočtů, protože X bývá typicky nula. Tím se také významně snižuje množství diskových I/O operací.

Tuto techniku jsem používal ve svém antivirovém programu. Každý řetězec antiviru Pasteur vyžadoval pouze jediný fixní začátek, konec a konstantní velikost. Byly podporovány i zástupné znaky, ale pouze v omezené míře. Řetězce byly setříděny do několika tabulek, v závislosti na typu objektu. Při porovnávání řetězců se vybral první bajt vstupního bodu a zkontrolovalo se, zda-li začíná jako některý řetězec prostřednictvím hašovacího vektoru. Jestliže nebyly nalezeny žádné takové první bajty, vybíraly se další vstupní body (pokud nějaké zbývaly).

Protože je velikost každého řetězce konstantní, algoritmus může také kontrolovat, zda-li se poslední bajt řetězce shoduje s právě skenovaným místem v souboru. Jestliže se poslední bajt řetězce shoduje, pak se shoduje celý řetězec. V praxi k tomu ovšem dochází zřídka. Tento trik je podobný algoritmu Boyer-Moore zkombinovaným s jednoduchým hašováním.

11.1.9 Hyper-rychlý přístup k disku

Hyper-rychlý přístup k disku je další užitečná technika, která se objevila v prvních implementacích antivirových skenerů. Používal jej program TBSCAN, stejně jako maďarský skener VIRKILL. Tyto skenery optimalizují proces skenování obcházením API určeného čtení z disku na úrovni operačního systému a přistupováním přímo přes BIOS. Protože byl MS-DOS obzvlášť pomalý při práci se souborovým systémem FAT, bylo možné dosáhnout nahrazením volání DOSových funkcí za přístup pomocí BIOSu až desetinásobného zrychlení při I/O operacích. Navíc byla tato technika užitečná jako obrana proti

stealth virům. Protože souborové stealth infektory obvykle obcházely pouze přístup k souborům na úrovni DOSu, bylo většinou možné vidět změny v souborech prostřednictvím volání BIOSu. Jiné skenery a programy pro ověřování integrity dat kvůli ještě vyšší rychlosti a bezpečnosti obcházely i BIOS a pracovaly přímo s řadiči disků.

Bohužel – dnes už není možné tyto techniky jednoduše (pokud vůbec) používat na všech operačních systémech. Ne jenom kvůli tomu, že existuje mnoho systémů souborů, které by antivirové skenery musely rozeznat a podporovat, ale také kvůli tomu, že existuje velký počet řadičů disku, což činí tuto úlohu prakticky nemožnou.

11.2 Skenery druhé generace

Skenery druhé generace používají přesnou a téměř přesnou identifikaci, která pomáhá vylepšit detekci počítačových virů a jiných škodlivých programů.

11.2.1 Chytré skenování

Chytré skenování se objevilo v době, kdy se objevily první počítačové viry s mutačními enginy. Takové enginy obvykle pracovaly se zdrojovými kódy assembleru a snažily se tam vkládat nadbytečné, nic-nedělající instrukce NOP. Rekompilovaný virus byl pak velmi odlišný od svého originálu, protože se v něm změnilo mnoho offsetů.

Chytré skenování přeskakuje v hostitelském programu instrukce typu NOP a neukládá takové instrukce ve virových signaturách. Snahou chytrého skenování je vybrat oblasti těla viru, které nemají žádné reference na data nebo další subrutiny. To zvýšilo pravděpodobnost detekce blízké varianty viru.

Tato technika je také užitečná při práci s počítačovými viry, které se objevují v textových formách, jako jsou třeba skriptovací a makro-viry. Tyto počítačové viry se umějí jednoduše změnit, protože mohou obsahovat prázdné znaky (jako např. mezery, nové řádky, tabulátory atd.). Tyto prázdné znaky se mohou při chytrém skenování odstranit ze skenovacích bufferů, čímž se významně zvýší detekční schopnosti takových antivirových skenerů.

11.2.2 Detekce struktury

Detekce struktury byla vyvinuta Eugenem Kasperskym a je hlavně užitečná při detekci rodin makrovirů. Namísto vybrání jednoduchého řetězce nebo kontrolního součtu sady maker skener raději zpracuje makra řádek po řádku a vypustí všechny nedůležité příkazy, stejně jako výše zmíněné prázdné znaky. Výsledkem je struktura těla makra, která obsahuje pouze nezbytný škodlivý kód, který se běžně objevuje v makrovirech. Skener pak tyto informace použije k detekci virů, čímž se dosáhne lepší detekce variant virů spadající do stejné rodiny.

11.2.3 Téměř přesná identifikace

Téměř přesná identifikace se používá pro přesnější detekci počítačových virů. Například – místo jednoho detekčního řetězce se pro každý virus použijí řetězce dva. K téměř přesné detekci viru Stoned můžeme vybrat z předchozího disassemblovaného výpisu druhý řetězec z offsetu 0x7CFC:

```
0700 BA80 00CD 13EB 4990 B903 00BA 0001
```

V případě, že skener detekuje jeden řetězec varianty viru Stoned, nepovolí odstranění viru, protože by se mohlo jednat o neznámou variantu, kterou by nemusel zvládnout korektně odstranit. Pokud jsou ale nalezeny dva řetězce, je virus téměř přesně identifikován. Sice se může stále jednat o nějakou variantu viru, ale samotné léčení má vyšší šanci na úspěch. Tato metoda je zejména bezpečná v případě, kdy je zkombinována se záložkami (bookmarks).

Jiná metoda téměř přesné identifikace je založena na použití kontrolního součtu (jako třeba CRC32) z vybrané oblasti z těla viru. Obvykle se vybere dezinfekční oblast těla viru a spočítá se její kontrolní součet. Výhodou této metody je lepší přesnost. To proto, že je možné vybrat v těle viru delší oblast, bez toho aniž by byla přetížená antivirová databáze – počet bajtů k uložení do databáze bude obvykle stejný jak pro malou, tak i pro velkou oblast, což se tak neděje v případě řetězců, protože delší řetězce zaberou více místa na disku a v paměti.

Skenery druhé generace dále mohou dosáhnout téměř přesné identifikace bez použití jakýchkoliv vyhledávacích řetězců, třeba za pomoci kryptografických kontrolních součtů[8] nebo nějakého druhu hašovací funkce.

Aby byly skenovací enginy rychlejší, začaly používat nějaký druh haše. To vedlo k tomu, že haš vypočítaný z kódu viru, nahradil detekci založenou na vyhledávacích řetězcích. Například antivirový skener Fridrika Skulasona pojmenovaný jako F-PROT[9] používá k detekci virů hašovací funkci společně se záložkami.

Další skenery druhé generace, jako třeba ruský KAV, nepoužívají žádné vyhledávací řetězce. Algoritmus KAVu vymyslel Eugene Kaspersky. Namísto řetězců používá skener dva kryptografické kontrolní součty, které jsou v rámci objektu vypočítané ve dvou předvolených pozicích a délce. Antivirový skener interpretuje databázi kryptografických kontrolních součtů, řadí data do skenovacích bufferů (v závislosti na formátu objektu) a porovnává součty se zařazenými daty. Buffer může například obsahovat kód ve vstupním bodě spustitelného souboru. V takovém případě se každý první kryptografický kontrolní součet, který koresponduje s detekcí kódu ve vstupním bodě, skenuje výpočtem prvního a druhého součtu. KAV zobrazí varování o možné variantě škodlivého kódu pouze v případě, že se shoduje první součet. Pokud se shodují oba kryptografické kontrolní součty, skener zobrazí varování s téměř přesnou identifikací. První rozsah součtu se obvykle optimalizuje tak, aby popisoval malou část těla viru, druhý bývá větší, aby pokryl jeho větší část.

11.2.4 Přesná identifikace

Přesná identifikace[9] je jediný způsob, jakým lze zaručit správnou identifikaci jednotlivých variant viru. Tato technika se obvykle kombinuje s technikami první generace. Narozdíl od téměř přesné identifikace, která používá kontrolní součty nějakého rozsahu bajtů v těle viru, přesná identifikace používá pokud možno co největší počet rozsahů, který je nezbytný k vypočítání kontrolního součtu všech jeho konstantních bitů. K dosažení této úrovně přesnosti se musí eliminovat proměnlivé bajty (variable bytes), aby se mohla vytvořit mapa všech konstantních bajtů. Konstantní data se dají v mapě použít, nicméně proměnlivé bajty mohou poškodit kontrolní součet.

Uvažte kód a data získaná z viru Stoned, která jsou zobrazena na obrázku 11.5. Na začátku kódu, v nultém bajtu těla viru, je možné nalézt dvě instrukce skoku, které nakonec převedou tok vykonávání programu na opravdový začátek virového kódu.

```
seg000:7C00                     bodyzero:
seg000:7C00 EA 05 00 C0 07              jmp     far ptr 7C0h:5
seg000:7C05 E9 99 00                    jmp     start
seg000:7C05                     ; ---------------------------------------------------------
seg000:7C08 00                  flag    db    0 ;               ; hard disk or diskette?
seg000:7C09 51 02               int13off dw 251h                ; DATA XREF: seg000:7CAF↓w
seg000:7C0B 00 C8               int13seg dw 0C800h              ; DATA XREF: seg000:7CB5↓w
seg000:7C0D E4 00               jumpstart dw 0E4h               ; offset to make
seg000:7C0D                                                     ; inter segment jump
seg000:7C0F 80 9F               virusseg dw 9F80h               ; DATA XREF: seg000:7CC6↓w
seg000:7C11 00 7C               bootoff dw 7C00h                ; to boot
seg000:7C13 00 00               bootseg dw 0                    ; segment
seg000:7C15                     ; ---------------------------------------------------------
seg000:7C15 1E                          push    ds
seg000:7C16 50                          push    ax
```

Obrázek 11.5 *Proměnná data viru Stoned.*

Těsně za druhou instrukcí skoku se nachází datová oblast viru. Proměnné jsou flag, int13off, int13seg a virusseg. Jedná se o skutečné proměnné, jejichž hodnoty se mohou lišit v závislosti na prostředí viru. Konstanty jsou jumpstart, bootoff a bootseg – tyto hodnoty se nemění, stejně jako zbytek virového kódu.

Protože všechny proměnlivé bajty jsou identifikovány, zbývá poslední důležitá věc ke zkontrolování – velikost virového kódu. Víme, že se Stoned vleze do jediného sektoru, nicméně virus se sám kopíruje do existujících boot a master boot sektorů. Pro nalezení skutečné velikosti viru je zapotřebí se podívat na kód, který se stará o kopírování viru do virového segmentu. Ten se dá nalézt v disassemblovaném výpisu na obrázku 11.6.

```
seg000:7CD3 B9 B8 01            mov     cx, 1B8h        ; 440 bytes
seg000:7CD6 0E                  push    cs
seg000:7CD7 1F                  pop     ds
seg000:7CD8 33 F6               xor     si, si          ; copy virus code to memory
seg000:7CDA 8B FE               mov     di, si
seg000:7CDC FC                  cld
seg000:7CDD F3 A4               rep movsb
seg000:7CDF 2E FF 2E 0D 00      jmp     dword ptr cs:0Dh
```

Obrázek 11.6 *Hledání velikosti těla viru Stoned (440 bajtů).*

Velikost viru je skutečně 440 (0x1B8) bajtů. Jakmile virus zkopíruje svůj kód do alokované paměťové oblasti, tam ihned předá řízení vlastnímu kódu. K tomu virus používá konstantní offset a virový segment virusseg uložený v datové oblasti na adrese CS:0Dh (0x7C0D). Nyní tedy máme všechny informace, které potřebujeme ke spočítání mapy viru.

Mapa bude zahrnovat následující rozsahy: 0x0-0x7, 0xD-0xE, 0x11-0x1B7 s kontrolním součtem 0x3523D929. Proměnlivé bajty jsou tedy eliminovány a virus je přesně identifikovatelný.

Abyste přesnou identifikaci lépe pochopili, uvažujme části kódu dvou variant viru Stoned, A a B, jak je vidět ve výpisech 11.1 a 11.2. Tyto dvě varianty mají stejnou mapu, takže jejich kód a konstantní oblasti dat se shodují. Kontrolní součet dvou různých variant se nicméně bude lišit, protože autor změnil pár bajtů ve zprávě a textové oblasti těla viru. Změna ve třech bajtech má za výsledek rozdílný kontrolní součet.

Výpis 11.1

Mapa viru Stoned.A.

```
Jméno viru: Stoned.A
Mapa viru: 0x0-0x7 0xD-0xE 0x11-0x1B7
Kontrolní součet: 0x3523D929
0000:0180 0333DBFEC1CD13EB C507596F75722050 ..........Your P
0000:0190 43206973206E6F77 2053746F6E656421 C is now Stoned!
0000:01A0 070D0A0A004C4547 414C495345204D41 .....LEGALISE MA
0000:01B0 52494A55414E4121 0000000000000000 RIJUANA!........
```

Výpis 11.2

Mapa viru Stoned.B.

```
Jméno viru: Stoned.B
Mapa viru: 0x0-0x7 0xD-0xE 0x11-0x1B7
Kontrolní součet: 0x3523C769
0000:0180 0333DBFEC1CD13EB C507596F75722050 ..........Your P
0000:0190 43206973206E6F77 2073746F6E656421 C is now stoned!
0000:01A0 070D0A0A004C4547 414C495A45004D41 .....LEGALIZE.MA
0000:01B0 52494A55414E4121 0000000000000000 RIJUANA!........
```

Přesná identifikace tak může od sebe odlišit obě varianty. Taková úroveň přesnosti je ovšem používána ve velmi malém množství AV produktů, jako třeba u antiviru F-PROT[9]. Přesná identifikace má jak pro koncové uživatelem, tak i pro antivirové výzkumníky mnoho výhod. Na druhou stranu – skenery používající přesnou identifikaci jsou při skenování infikovaných systémů obvykle pomalejší než jednoduché skenery (pokud jsou jejich algoritmy pro přesnou identifikaci použity).

V případě většího počítačového viru je také docela únavné mapovat všechny konstantní rozsahy. To proto, že virový kód často míchá data společně s kódem.

11.3 Algoritmické skenovací metody

Algoritmické skenování je trochu matoucí, nicméně široce používaný termín. Jakmile si standardní algoritmy skeneru nedovedou s virem poradit, je zapotřebí implementovat nový, pro daný virus specifický detekční algoritmus. Tomu se říká algoritmické skenování, nicméně termín *algoritmus detekce konkrétního viru* zní mnohem jasněji. První implementace algoritmického skenování byly řešeny jako obyčejné sady pevně daných detekčních rutin, které se přikládaly k samotnému kódu jádra antivirového enginu.

Není překvapením, že takový typ detekce způsoboval mnoho problémů. První spočíval v tom, že kód enginu byl smíchán se speciálními rutinami, které se obtížně programovaly na nové platformy. Druhý problém tkvěl v ohrožení stability – algoritmické skenování mohlo skener jednoduše shodit, protože aktualizace pro detekce virů se vždy musí vypouštět co nejdříve, přičemž není dost času na testování.

Řešení tohoto problému je jazyk na skenování virů[10]. Takové jazyky v té nejjednodušší formě obvykle podporují operace posunu ukazatele a čtení ve skenovaných objektech. Algoritmický sken podle řetězce se tedy provede přesunutím ukazatele na příslušné místo směrem dopředu od začátku souboru (nebo dozadu od jeho konce nebo od vstupního bodu), čtením bajtů, spočítáním místa, kam volání ukazuje a porovnáním řetězců, pěkně jednoho po druhém.

Algoritmické skenování je nezbytnou součástí architektury moderních antivirových systémů. Některé skenery, jako třeba KAV, přišly s objektovým kódem, který byl součástí antivirové databáze. Detekční rutiny pro jednotlivé viry jsou napsány v přenositelném jazyce C, zkompilovány do objektového kódu a uloženy v databázi skeneru. Skener implementuje zavaděč podobný tomu z operačního systému, který za běhu spojí všechny objekty zodpovědné za detekci virů. Ty se potom spustí jeden po druhém, v závislosti na předdefinovaném pořadí volání. Výhodou této implementace algoritmického skenování je lepší výkon, nevýhodou je možné riziko nestability způsobené spouštěním reálného kódu na běžícím systému, který může obsahovat drobné chyby způsobené rychlou reakcí antivirových společností na hrozící nebezpečí, které si vyžádalo rychlé vytvoření komplexní detekční rutiny.

K eliminaci tohoto problému se moderní algoritmické skenování implementuje jako p-kód (portable code, přenositelný kód), který je podobný Javě, s použitím virtuálního stroje. Příkladem použití této techniky je Norton AntiVirus. Výhodou této metody je to, že detekční rutiny jsou vysoce přenositelné. Není totiž potřeba vytvářet znovu pro nové platformy každou detekční rutinu specifickou pro daný virus – ty totiž mohou běžet stejně dobře na PC jako na IBM AS/400 (v případě, že je skener a virtuální stroj algoritmického skenovacího enginu přeportován na takové platformy). Nevýhodou je relativně pomalé vykonávání p-kódu (v porovnání s přímým spouštěním). Interpretovaný kód totiž obvykle běží až stokrát pomaleji než pravý strojový kód. Detekční rutiny mohou být implementovány v jazyce assembleru s vysokoúrovňovými makry. Takové rutiny pak mohou poskytovat skenovací funkce, které vyhledávají skupiny řetězců. Takové skenery ovšem musí být optimalizovány filtrováním, které je více rozvedeno v následující části. Detekční kód může být také implementován v rozšiřitelném skenovacím enginu za použití nativního kódu.

V budoucnu se očekává, že algoritmické skenery budou implementovat JIT (Just-In-Time) pro kompilaci detekčních rutin založených na p-kódu do nativního kódu dané architektury, podobně jako to dělá .NET Framework v Microsoft Windows. Například, když se skener spustí na platformě Intel, p-kód se za běhu zkompiluje do kódu Intelu, čímž se rychlost vykonávání p-kódu dramaticky zvýší (často více než stonásobně). Tato metoda pak eliminuje problémy při spouštění strojového kódu jako součásti databáze, přičemž samotné vykonávání zůstává pod kontrolou skeneru, protože detekční rutiny se skládají z řízeného kódu (managed code).

11.3.1 Filtrování

Filtrovací technika se stále více používá ve skenerech druhé generace. Myšlenka filtrování spočívá v tom, že viry obvykle infikují pouze vybranou sadu známých typů objektů, což dává skeneru jistou výhodu. Například signatury boot virů se mohou omezit pouze na boot sektory, signatury DOSových EXE infektorů na soubory EXE atd. U řetězce nebo detekční rutiny se právě z tohoto důvodu používá speciální příznak, který indikuje, zda-li se má signatura v právě skenovaném objektu kontrolovat, což redukuje množství řetězců, které skener musí následně porovnat.

Algoritmické skenování silně závisí na filtrech. Protože jsou takové detekce více závislé na výkonu skenování, algoritmická detekce vyžaduje filtrování na dobré úrovni. Filtrem může být cokoliv, co je specifické pro daný virus: typ spustitelného souboru, identifikační značka viru v hlavičce skenovaného objektu, sekce kódu s podezřelými příznaky nebo jmény atd. Bohužel – ne každý virus umožňuje použít proces filtrování.

Problém skenerů je jasný – skenování takových virů je časově náročné pro všechny produkty. Další problémem je detekce vyvíjejících se virů (jako třeba zakódované a polymorfní viry), které se dají najít prostřednictvím řetězců se zástupnými znaky pouze ve výjimečných případech.

Tyto viry se dají lépe detekovat s použitím generického dekryptoru[11] (založeném na virtuálním stroji), který dekóduje tělo viru a nalezne jeho konstantní tělo za použití řetězců nebo jiné známé metody detekce. Tyto metody nicméně nejsou vždy funkční. Například takové EPO viry a viry s obranou proti emulaci jsou schopny tyto techniky odstavit. V takových případech je zapotřebí použít zcela jiné techniky, jako je třeba analýza decryptoru/polymorfního enginu. Takto se dají dokonce detekovat i viry jako třeba W32/Gobi[12] (analýza decryptoru jednoduše znamená, že se nahlédne do několika polymorfních decryptorů a porovnají se části kódu decryptoru v polymorfním enginu – takto se dá v mnoha případech, za použití algoritmické detekce, detekovat i samotný decryptor).

Kód algoritmické detekce obvykle stráví mnoho času ve smyčce, což vyžaduje značný výkon procesoru. Například, v některých případech vyžaduje vysoce optimalizovaná detekce viru W95/Zmist[13] vykonání přes dva milióny iterací p-kódu, aby byl virus korektně rozpoznán. Tento druh detekce funguje pouze tehdy, když se dá infikovaný soubor rychle rozeznat a odlišit od souboru neinfikovaného.

Ačkoliv varianty viru Zmist nijak neoznačují napadené objekty, existuje přesto několik způsobů, jakým lze napadené soubory odfiltrovat. Zmist totiž používá několik filtrů, aby omezil infekci některých spustitelných souborů. Například infikuje pouze takové soubory, které mají známá jména sekcí a neinfikuje soubory nad stanoveným limitem jejich velikosti. Kombinace takových filtrů redukuje množství souborů nutných pro zpracování na méně než 1% všech spustitelných objektů, což umožňuje, aby relativně náročný detekční algoritmus běžel efektivně na všech systémech.

Pro efektivní filtrování se dají použít následující kontroly:

- Kontrola počtu nulových bajtů v oblasti souboru, kde se předpokládá umístění viru. Ačkoliv některé viry používají kódování, četnost zakódovaných a nezakódovaných dat se může dost lišit. Taková technika se běžně používá pro kryptografické prolamování. Například konec PE souboru (posledních pár kilobajtů) často obsahuje více než 50% nulových bajtů. V zakódovaném viru bývá obvyklý výskyt nul mnohem menší, pod 5 procent.
- Kontrola změn příznaků a velikostí v hlavičce sekce. Některé viry označují sekce jako zapisovatelné a jiné podobným způsobem mění některé důležité záznamy na atypické hodnoty.
- Kontrola příznaků souboru. Některé viry neinfikují konzolové aplikace, jiné neinfikují DLL knihovny nebo systémové ovladače.

11.3.2 Statická detekce decryptoru

Problémy se objevují v okamžiku, kdy je tělo viru náhodně zakódované, protože rozsahy bajtů, které skener používá k identifikaci viru, jsou omezené. Různé produkty používají detekci decryptoru specifickou pro daný virus všemi způsoby, ve všech kódových sekcích programových souborů. Rychlost skenování závisí na velikosti kódových sekcí skenovaných aplikací. Taková metoda detekce se používala ještě předtím, než se objevily první generické decryptory. Sama o sobě může tato technika hlásit falešné poplachy a opomíjet zavirované soubory, nehledě na to, že nezajišťuje možnost léčení, protože kód viru není dekódován. Jakmile se ale tato metoda použije společně s nějakým efektivním filtrem, je relativně docela rychlá.

Podívejte se na část kódu viru W95/Mad, který je uveden na obrázku 11.7, který je dále rozebrán v kapitole 7. Decryptor viru W95/Mad je na počátku zakódovaného těla, hned za vstupním bodem infikovaných PE souborů.

```
00404200                            public start
00404200                          start:
00404200 E8 00 00 00 00             call $+5
00404205 5F                         pop  edi
00404206 8B C7                      mov  eax, edi
00404208 2D 05 22 00 00             sub  eax, 2205h       ; variable intruction operand!
0040420D
0040420D                          init:                   ; CODE XREF: .reloc:00404283↓j
0040420D 81 EF 05 30 40 00          sub  edi, 403005h
00404213 89 87 3C 30 40 00          mov  [edi+40303Ch], eax ; entry of host
00404219 89 AF 40 30 40 00          mov  [edi+403040h], ebp ; saved for later use
0040421F 8B EF                      mov  ebp, edi
00404221 33 C0                      xor  eax, eax
00404223 BF 45 30 40 00             mov  edi, 403045h
00404228 03 FD                      add  edi, ebp         ; adjust EDI to "decrypted"
0040422A B9 6B 0A 00 00             mov  ecx, 0A6Bh       ; number of bytes to decrypt
0040422F 8A 85 44 30 40 00          mov  al, [ebp+403044h] ; pick key (constant byte)
00404235
00404235                          decrypt:                ; CODE XREF: .reloc:00404238↓j
00404235 30 07                      xor  [edi], al        ; decrypt current byte
00404237 47                         inc  edi              ; position to next byte
00404238 E2 FB                      loop decrypt
0040423A EB 09                      jmp  short near ptr decrypted
0040423A                          ; ---------------------------------------------------------
0040423C 00 10 40 00                dd 401000h            ; stored host EP
00404240 78 FF 63 00                dd 63FF78h            ; stored EBP
00404244 7B                       key db  7Bh ; {
00404245 BD                       decrypted db 0BDh ; +   ; CODE XREF: .reloc:0040423A↑j
00404246 FE                         db 0FEh ; ■
00404247 28                         db  28h ; (
00404248 42                         db  42h ; B
00404249 3B                         db  3Bh ; ;
0040424A 7B                         db  7Bh ; {
0040424B 7B                         db  7Bh ; {
```

Obrázek 11.7 *Decryptor viru W95/Mad.*

V tomto příkladu je operand instrukce SUB umístěný na 404208 proměnlivý, takže je zapotřebí ve vstupním bodě použít řetězec se zástupnými znaky. Následující řetězec bude schopný tento decryptor detekovat i v případě jiných variant viru:

```
8BEF 33C0 BF?? ???? ??03 FDB9 ??0A 0000 8A85 ???? ???? 3007 47E2 FBEB
```

Protože tento virus používá k dekódování svého těla jednoduchou metodu (XOR s konstantním bajtem), kompletní dekódování virového kódu je vcelku prosté. Dekódování se dá docílit velmi jednoduše, protože délka klíče není příliš velká. V našem příkladu je klíč 7Bh. Všimněte si těchto bajtů v zakódované oblasti viru – jedná se o nulové bajty, protože 7Bh XOR 7Bh = 0. Protože známe konstantní kód pod

jednou zakódovanou vrstvou, je útok na čistý (nezakódovaný, dekódovaný) text snadno proveditelný. Tato metoda detekce je předmětem následující části kapitoly.

Detekce decryptoru se dá použít i pro detekci polymorfních virů. I velmi silné mutační enginy, jako třeba MtE, používají alespoň jeden konstantní bajt v decryptoru, což stačí pro algoritmickou detekci s použitím disassembleru délky instrukcí. Polymorfní decryptor se jím dá disassemblovat a kód decryptoru profilovat. MtE používá konstantní instrukce podmíněného skoku zpět na proměnlivém místě. Operační kód instrukce je 75h, což se dekóduje jako instrukce JNZ. Operand instrukce vždy ukazuje směrem zpět, což značí, že se jedná o decryptor. Potom se tok programu zanalyzuje na všechny možné způsoby, kterými virus dekóduje své tělo, přičemž se ignorují všechny nadbytečné instrukce.

Je sice časově náročné analyzovat vylepšené polymorfní enginy pro všechny možné dekódovací metody a nadbytečné instrukce, ale často je to jediný způsob, jakým lze takové viry detekovat.

11.3.3 Rentgenová metoda (X-raying)

Jiná skupina skenerů používá kryptografickou detekci. Jak lze vidět v předchozím zmíněném příkladu, virus W95/Mad používá konstantní zakódování přes XOR s náhodně zvoleným bajtem, který slouží jako (de)kódovací klíč uložený v těle viru. To činí dekódování a následnou detekci viru opravdu triviální záležitostí. Podívejte se na část těla viru W95/Mad v zakódovaném a dekódovaném tvaru ve výpisech 11.3 a 11.4. V následujícím příkladě slouží jako decryptovací klíč bajt 7Bh.

Výpis 11.3

Zakódovaná část viru W95/Mad.

```
Zakódovaný text                         ASCII bajty
5B4A42424C7B5155 - 1E231E7B20363A3F     [JBBL{ QU.#.{  6:?
5B1D14095B2C1215 - 424E265B0D1E0908     [...[,..BN&[....
1214155B4A554B5B - 393E2F3A5A5B5318     ...[JUK[9>/:Z[S.
5239171A18105B3A – 151C1E171B424C7B     R9....[:.....BL{
2031393937002A2E - 655865005B4D4144     1997.*.eXe.[MAD
20666F722057696E - 39355D2076657273     for Win95] vers
696F6E20312E3020 - 4245544121202863     ion 1.0 BETA! (c
29426C61636B2041 - 6E67656C60393700)    Black Angel`97.
```

Výpis 11.4

Dekódovaná část viru W95/Mad.

```
Odpovídající čistý text                 ASCII bajty
2031393937002A2E - 655865005B4D4144     1997.*.eXe.[MAD
20666F722057696E - 39355D2076657273     for Win95] vers
696F6E20312E3020 - 4245544121202863     ion 1.0 BETA! (c
29426C61636B2041 - 6E67656C60393700     )Black Angel`97.
```

Virus se dá jednoduše dekódovat algoritmickou technikou a následně přesně identifikovat.

Antiviroví výzkumníci také mohou prozkoumat polymorfní enginy vylepšených virů a vypátrat metody zakódování, které používají. Jednoduché metody, jako třeba XOR, ADD, ROR atd. se často používají s 8bitovými, 16bitovými a 32bitovými klíči. Někdy decryptor viru používá více než jednu vrstvu, která je zakódována stejnou metodou (nebo i více metodami) s jednobajtovým, dvoubajtovým a čtyřbajtovým klíčem.

Útok na zakódování virového kódu se nazývá jako rentgenové (X-RAY) skenování, které vymyslel Frans Veldman pro svůj produkt TBSCAN, stejně jako někteří další výzkumníci v podobném čase a nezávisle na sobě. Poprvé jsem tuto techniku použil pro detekci viru Tequila. Myšlenka rentgenování byla vcelku přirozená, protože dekódování kódu viru bylo – pro účely léčení – nezbytné i v případě starších známých, a neustále se šířících virů, jako třeba Cascade. Veselin Bontchev mě informoval, že poprvé spatřil dokument popisující techniku rentgenování u Eugena Kasperskyho.

Rentgenové skenování využívá všech jednoduchých metod kódování a provádí se na vybraných oblastech v souborech, jako třeba na jeho začátku, na konci nebo blízko kódu u vstupního bodu programu. Skener tedy stále může pro detekci zakódovaných – a dokonce i složitých polymorfních – virů[14] používat jednoduché řetězce. Skenování je sice trochu pomalejší, ale technika je obecně použitelná.

Problém s touto metodou vyvstane v momentě, kdy začátek těla viru není k nalezení na fixním místě a útok proti decryptoru se tak musí spustit na velké oblasti souboru, což vede k velkému zpomalení. Výhodou této metody je kompletní dekódování virového kódu, což umožňuje soubor vyléčit i v případě, kdy jsou informace, které jsou nezbytné k obnovení, uloženy v zakódované formě.

Poznámka

Rentgenové skenování často detekuje instance počítačových virů, které mají falešný decryptor. Některé polymorfní viry vytvářejí falešné decryptory, které nedekódují tělo viru správně, nicméně dekódování rentgenovou metodou skončí i na těchto vzorcích úspěchem. Takové vzorky se dají detekovat pouze tímto způsobem, protože detekční techniky založené na emulaci, vyžadují fungující decryptor.

Některé viry, jako třeba W95/Drill, používají více než jednu zakódovanou vrstvu a polymorfní engine, ale i tak mohou být efektivně detekovatelné. Nejvíce záleží na kombinaci kódovacích metod. Příliš nezáleží na tom, jestli polymorfní zakódování používá metodu XOR jednou nebo stokrát – v obou případech se dá virus rentgenováním detekovat. Nebo třeba takový polymorfní engine SMEG (Simulated "Metamorphic" Encryption Generator), vytvořený autorem jménem Black Baron, se dá efektivně detekovat s použitím algoritmické detekce, která využívá techniku rentgenování.

Poznámka

Christopher Pile, autor virů SMEG, byl v listopadu roku 1995, podle zákona Computer Misuse Act Spojeného Království, odsouzen k osmnácti měsícům ve vězení.