Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar

Vírusok, férgek szerepe az informatikai hadviselésben

Szerző:

Adamkó Péter

Konzulens:

Szigeti Szabolcs, Krasznay Csaba

Tartalomjegyzék

1. Előszó

Maguk a vírusok, mintegy 30 éve jelen vannak életünkben, bár jelentőségüket és hatásukat csak az utóbbi pár évben érezhetjük át igazán, az internet és a számítógépes világ széles körű elterjedésével. Sokan úgy gondolhatják, hogy napjainkban a vírusok kérdése nem túl érdekes téma, mára már majdnem minden lehetséges védekezési és támadási módot láthattunk. A felhasználók tudatosságának mélyítése elegendő lesz a jelenlegi eszközök és módszerek mellett, hogy biztonságban lehessünk. Véleményem szerint ez nem így van. A vírusok történelme igen tanulságos, s mutatja, évről-évre találkozhatunk újszerű megoldásokkal, ötletekkel ezen a fronton is. Ráadásul nem számíthatunk arra, hogy az, aki számítógépet használ, az érteni is fog hozzá, hiszen ehhez nem kell jogosítvány, vagy képesítés. De tekintsük el egy kicsit ettől. Az már régóta bizonyított tény, hogy egy vírus fertőzésének kimutatása visszavezethető a Turing gép megállási problémájára. Mivel ez a probléma algoritmikusan eldönthetetlen, így a vírus detektálása is az (Adleman - 1988). Természetesen vannak eszközeink, amik lehetővé teszik, hogy nagy részüket kiszűrjük, azonban a fejlődés nem áll meg!

A történeti áttekintés, valamint a rendelkezésre álló módszerek technikák ismertetése révén, előretekintést kívánok adni a jövő fejlődési, kutatási irányairól. A vírusírók és vírusírtók párharca mindig is adok-kapok játék volt. Amennyiben az egyik oldal valamilyen új dologgal jelentkezett, akkor a másik annak megkerülésével próbálkozott azonnal. Például, amint elterjedtek a polimorf motorral rendelkező vírusok, létrejöttek a heurisztikákat használó védelmi szoftverek, majd ezen alapulva létrejöttek a kódolt vírusok.

A technológiai leírások során megmutatom, milyen lehetőségeket ad az operációs rendszer, a fájl formátumok, és a hálózati protokollok felépítése a támadók számára. Egyenként mindegyik eljárást észlelni lehet, azonban ötvözve ezeket a feladat nagy komplexitásúvá válik.

A mérés során bemutatom, hogy bár létjogosultsága a gyors férgeknek is van, hosszútávon a rootkit technológia és annak óvatos alkalmazása a hackerek, és más rosszindulatú felhasználók fő eszköze.

2. Vírustörténelem, technológiai fejlődés a kezdetektől napjainkig

2.1. A vírusok története

Az 1970-es évek

A Tenex operációs rendszer alatt kifejlesztett The Creeper vírus az ARPANET-en terjedt. Képes volt belépni modemen keresztül egy hálózatba és átküldeni önmaga egy másolatát a távoli rendszerre. A vírus ellen hozták létre a "The Reeper" nevű programot, amely törölte azt, hasonló technikát használva.

Az 1980-as évek elején

A számítógépek egyre elterjedtebbek lettek, már nem csak vállalatok írtak programokat, hanem felhasználók is, akik saját a programjaikat szabadon terjesztették és cserélték nyilvánosan hozzáférhető szervereken, a BBS-eken keresztül. Ennek eredményeként nagyszámú trójai program is megjelent.

1981 eseményei

Megjelent az Elk Cloner, mely az Apple II számítógépeken terjedő bootvírus. A vírus a floppy lemezek bootszektorát módosította, valahányszor csak elindították. Mivel akkoriban az operációs rendszer floppy lemezeken tárolódott, ez minden indulásnál megtörtént. Többféle viselkedése volt: átkapcsolta a képernyőt, villogtatta a szöveges képernyőt, különféle üzeneteket jelenített meg, és képeket forgatott.

1986 eseményei

Ekkor jelent meg az első IBM PC vírus. A pakisztáni Brain világszerte elterjedt az IBM kompatibilis PC-k körében. A vírus 360 KB-os kapacitású lemezeket fertőzte meg, szintén a lemezek bootszektorát módosította. A Brain volt az első lopakodó vírus, olvasási kérés esetén a vírus a fertőzött szektor helyett, annak eredeti tartalmát adta vissza, melyet valahova a lemezre mentett el.

Ugyancsak ebben az évben történt, hogy Ralph Burger elkészítette az első fájlvírust, mely képes volt saját kódját egy másik program végrehajtható fájljához kapcsolni. Ez volt a VirDem vírus.

1987 eseményei

Megjelent a Vienna vírus, mely „com” fájlok megfertőzésére volt képes. Ralph Burger visszafejtette a vírust, s ezt publikálta "Computer Viruses: a High-tech Disease" című könyvében. Ez a könyv népszerűvé tette a vírusírást, s bátorította sok vírus megalkotását, melyek alkalmazták a könyvből vett ötleteket.

Néhány további IBM-PC vírus is megjelent ebben az évben. A Lehigh a COMMAND.COM fájlt fertőzte meg, a Suriv-1 már az összes fajta COM fájlt képes volt megfertőzni. Az első EXE fájlt fertőző vírus a Suriv-2 volt, a Suriv-3 pedig már képes volt megfertőzni a COM, és EXE fájlokat egyaránt. Néhány újabb bootvírus is felbukkant, az USA-ban a Yale, Új-Zélandon a Stoned, Olaszországban a PingPong, valamint ebben az évben megjelent az első öntitkosító fájlvírus, a Cascade.

Más platformokról sem szabad megfeledkezni, vírusok jelentek meg Apple Macintosh, Commodore Amiga és Atari ST számítógépekre is.

1987 decemberében tört ki az első teljes hálózatot érintő járvány, amit a REXX nyelven írt Christmas Tree okozott, amely VM/CMS operációs rendszer alatt működött. A Bitnet hálózatából indulva kapugépeken keresztül a vírus bejutott az Európai Akadémiai Kutatási Hálózatba (European Academic Research Network, EARN) és onnan az IBM Vnet rendszerébe. Négy nappal később lebénította a hálózatot, amelyet túlterheltek a vírus másolatai. A vírus egy karácsonyfát rajzolt ki, majd elpostázta másolatait az összes hálózati címre a felhasználóknak, amelyeket a NETLOG és NAMES nevű rendszerfájlokban talált.

1988 eseményei

1988-ban pénteken 13-án a jelentkezett a Jerusalem vírus. Ezen a napon a vírus tönkretette azokat a fájlokat, amelyeket megpróbáltak futtatni. Talán ez volt az első olyan MS-DOS vírus, amely világszerte elterjedt. Fertőzésekről Európában, Amerikában és a Közel-Keleten is hallani lehetett. A Jerusalem néhány további vírussal együtt (Cascade, Stoned, Vienna) még mindig észrevétlenül tevékenykedett, számítógépek ezreit fertőzve meg, mivel a vírusellenes programok akkoriban még nem nagyon voltak elérhetők. Sok felhasználó, sőt szakemberek sem hittek a számítógépvírus létezésében.

Rendszeresen kezdtek megjelenni különféle hamis vírusokról szóló rémhírek, pánikot keltve a felhasználók között.

Egy nagyméretű hálózati járványt okozott a Morris (más néven Internet Worm) féreg. A vírus több mint 6000 számítógéprendszert fertőzött meg az USA-ban, beleértve a NASA kutatóintézetét is, és gyakorlatilag teljesen megbénította azokat. A vírus végtelen számú másolatot küldött szét magából a hálózaton (mint a Christmas Tree nevű féreg), és erre nem volt elég a megfertőzött rendszerekben rendelkezésre álló hálózati erőforrása kapacitás. A Morris féreg mintegy 96 millió dolláros kárt okozott.

A vírus a VAX és a Sun Microsystems Unix rendszereinek hibáját kihasználva terjedt (sendmail). A Unix hibája mellett a vírus néhány egyéb akciót is végrehajtott, például összegyűjtötte a felhasználói jelszavakat.

Hamarosan újabb férgek jelent meg, a DECNet hálózatán. A HI.COM nevű féreg egy lucfenyő képét rajzolta ki és azt közölte a felhasználókkal, hogy "stop computing and have a good time at home!”. Néhány új antivírus program is feltűnt, mint például a Doctor Solomon's Antivirus Toolkit.

1989 eseményei

Új vírusok jelentek meg, mint a Datacrime, a Fu Manchu, és egész víruscsaládok is, mint a Vacsina és a Yankee Doodle. Az első rendkívül pusztító volt, mivel október 13. és december 31. között leformázta a merevlemezeket. Ez a vírus már nem csak kutató-laboratóriumokban fordult elő, hanem az egész világon, így sokkal nagyobb sajtóvisszhangot kapott, főleg Hollandiában és Nagy-Britanniában.

Októberben újabb féreg terjedt el a DECNet-en, a Wank Worm.

Decemberben újabb incidens történt, 20,000 lemezt kézbesítettek ki "AIDS Information Diskette Version 2.0" címkével. 90 bootolás után a rajta levő trójai program lekódolta a lemezen lévő összes fájlnevet és rejtetté tette azokat. Mindössze egyetlen fájl maradt olvasható, egy számla, mely 189 dollárról szólt.

Ebben az évben Oroszországban is kiterjedt fertőzéseket okozott a Cascade, a Jerusalem és a Vienna. Az orosz programozók hamar visszafejtették a vírusokat, hamarosan antivírus programok jelentek meg, közöttük az AVP (AntiViral Toolkit Pro).

1990 eseményei

Az év elején megjelent a Chameleon, az első polimorfikus víruscsalád, melynek tagjai "V2P1", "V2P2" és "V2P6" néven váltak ismertté. Egészen eddig az antivírus programok

mintafelismerést használtak a víruskeresésre. A Chameleon megjelenése után az antivírus programok fejlesztőinek új módszereket kellett kitalálniuk.

A második jelentős esemény a „bolgár” vírusgyár megjelenése volt, nagy mennyiségben kerültek elő Bulgáriában írt vírusok. Nagyobb járványokat következő víruscsaládok okoztak: Murphy, Nomenclatura, Beast (más néven 512 vagy Number-of-Beast) és az Eddie vírus változatai.

A vírusgyár leghíresebb tagja, Dark Avenger, több olyan vírust is készített évente, amelyek alapvetően új algoritmusokat alkalmaztak a fertőzéshez és a vírus nehezebb észlehetőségének érdekében. Bulgária az első olyan ország, ahol megnyílt a kifejezetten víruskód és egyéb hasznos információk cseréjére specializálódott BBS.

Júliusában a "PC Today" nevű, Nagy-Britanniában megjelenő számítógép magazin floppy mellékletével terjedt a DiskKiller nevű vírus, melyből több mint 50 ezer példány került forgalomba.

Az év második felében két lopakodó technikát alkalmazó szörnyeteg is felbukkant, a Frodo és a Whale. Mindkét vírus rendkívül összetett algoritmust használt, sőt a 9 KB-os (ami meglehetősen nagy méretűnek számított akkor) Whale többszintű titkosító, és visszafejtést akadályozó technikákat is alkalmazott.

Megjelentek az első orosz vírusok: Peterburg, Voronezh, és a LoveChild.

Létrejött az EICAR (European Institute for Computer Antivirus Research).

1991 eseményei

A számítógépvírusok száma hatalmas sebességgel növekedett, már háromszáz fölé emelkedett az ismert vírusok száma. Ez év áprilisában egy kiterjedt vírusjárványt okozott a fájlfertőző és polimorf bootvírus tulajdonságokkal rendelkező Tequila, majd szeptemberben ugyanezt tette az Amoeba vírus. Nyáron a Dir2 okozott világszerte nagy pusztítást és kiterjedt járványt. Ez a vírus alapvetően új technikát alkalmazott a fertőzésre, a programfájlok belsejébe írt a kód helyett, a FAT link bejegyzéseit irányította magára.

1992 eseményei

Az év jelentős fordulatot hozott, szinte teljesen eltűntek azok a vírusok, amelyek nem IBM PC-ket és nem az MS-DOS rendszert fertőzték, ez is mutatja az operációs rendszer népszerűségét. Ezt a platformot fertőző fájl, boot, és ezt a két technikát kombináló vírusok

egyre jellemzőbbek lettek. A vírusok száma folyamatosan növekedett, szinte naponta történtek újabb és újabb incidensek.

Az év elején bukkant fel az első polimorf vírusok sorozatgyártását segítő programgenerátor, az MtE (Mutation Engine), Dark Avenger alkotása. A programgenerátort természetesen gyorsan elterjedt, s jó néhány polimorf vírusváltozat köszönhette neki születését. Szerencsére a generátorra épülő vírusok nagy része felismerhető volt, mivel ugyanazon közös rész volt meg bennük.

1992 márciusában a Michelangelo vírus nem csupán valóban kiterjedt, világméretű vírusjárványt okozott, de nagy visszhangot is a sajtóban. Ez volt az első eset, amikor egy vírus hatását jelentősen túlbecsülték a vírusirtó cégek, pánikot okozva ezzel.

Júliusban megjelentek az első víruskészítő programcsomagok, a VCL (Virus Construction Laboratory) és a PS-MPC. Ezek tovább növelték az új vírusok számát.

Az év végén jelentek meg az első Windows vírusok, amelyek a Windows végrehajtható rendszerfájljait fertőzték meg.

1993 eseményei

A víruskészítők már komoly károkat kezdtek okozni, emellett egyre több olyan vírus jelent meg, amelyek újszerű fájlfertőzési és bejutási módszereket alkalmaztak. A PMBS – Az első olyan vírus volt, amelyben Intel 80386 védett (protected) módú programozást alkalmaztak.

A Strange, vagy más néven Hmm lopakodó technikát használt, az INT 0Dh és INT 76h hardver megszakításokat használta fel.

Az Emmie, a Metallica, a Bomber, az Uruguay és a Cruncher szintén alapvetően új technikákat alkalmaztak arra, hogy elrejthessék saját, a fertőzött programba beírt programkódjukat.

1994 eseményei

A CD lemezeken terjedő vírusok száma ugrásszerűen nőtt. A lemezek hatalmas népszerűségre tettek szert, s elsődleges terjesztési médiaként a vírusterjesztés fontos eszközévé váltak.

Az év elején Nagy-Britanniában két rendkívül összetett polimorf vírus (az SMEG.Pathogen és az SMEG.Queeg) került a CD-kre.

Az év folyamán néhány meglehetősen szokatlan vírus is felbukkant, januárban jelent meg a Shifter nevű vírus, mely az első object modulokat (OBJ fájlokat) fertőző vírus volt. A Phantom1 vírus volt az első polimorf vírus okozta járvány, mely Moszkvában előfordult.

Áprilisban jelentkezett a SrcVir, mely a programok forráskódjaiba (C és Pascal nyelvű) írta bele saját kódját.

Júniusában jelent meg Szlovákiában a OneHalf, és végül szeptemberben a 3APA3A nevű bootvírus okozott járványokat, elsősorban Oroszországban.

1995 eseményei

Ebben az évben nem jelent meg semmilyen újszerű DOS vírus, bár néhány meglehetősen összetett programkód, mint a NightFall, a Nutcracker, és emellett néhány fura vírus, mint az RMNS vírus valamint a Winstart nevű BAT vírus előkerült. A BayWay és a DieHard2 vírusok szintén széles körben terjedtek el.

Februárban történt, hogy a Windows 95 béta verziójának lemezei Form vírussal fertőzötten kerültek ki a tesztelőkhöz.

1995 augusztusában bekövetkezett az eddigi legnagyobb fordulópont, megjelent a Microsoft Word dokumentumait felhasználó első "élő" vírus (a WM/Concept). Pár hónap alatt világszerte elterjedt, és hosszú hónapokig vezette a különböző számítógépes magazinok fertőzési statisztikáit. Az év végén újabb két, lemezen terjedő vírusnak lehetünk tanúi, ezek a PC Magazine (angol nyelvű verziójának), valamint a Computer Life karácsonyi üdvözletében voltak megtalálhatóak.

Technikai újításként megjelentek a makrovírusok.

1996 eseményei

Januárban megjelent az első Windows 95 vírus (a Win95/Boza), márciusban pedig tanúi lehettünk az első Windows 3.x vírus (Win/Tentacle) okozta járványnak.

Júniusban bukkant fel az OS2/AEP, az első OS/2-re írt vírus, mely jól fertőzi meg az operációs rendszer EXE fájljait. Idáig OS/2 alatt csak olyan vírusok léteztek, melyek vagy a megtámadott programot írták felül, vagy társvírusként mellé telepedtek.

Júliusban a Word után az Excel következett, megtalálták az első Excelt fertőző makróvírust, az XM/Laroux-t, két helyen egyidejűleg, egy alaszkai olajtársaságnál és a Dél-Afrikai Köztársaságban. A Laroux hasonlóan a Microsoft Word vírusaihoz, a dokumentumokban

elhelyezhető makrókon alapszik. Ilyen programokat mind az Excel elektronikus számolótábláiban, mind a Word dokumentumaiban el lehet helyezni. Nemsokára kiderült, a Microsoft Excelbe épített VisualBasic programnyelv is alkalmas vírusok létrehozására és futtatására.

Decemberben jelent meg a Win95/Punch, mely az első Windows 95-re írt memóriarezidens vírus. VxD meghajtóként maradt a Windows memóriában, s a fájl-hozzáférések figyelésével a Windows EXE fájlokat azok megnyitásakor fertőzte meg.

Általánosságban véve 1996-ban kezdődött meg a támadás a 32 bites Windows operációs rendszerek (Windows 95 és Windows NT) és alkalmazásaik (Office) ellen.

1997 eseményei

1997 februárjában megjelent az első Linuxra kifejlesztett vírus, a Linux.Bliss. A vírusok ismét egy új platformot kezdtek el támadni. Ezzel együtt Linuxon máig nem terjedtek el túlságosan, főleg az operációs rendszer kisebb népszerűsége miatt. Megjelentek a Microsoft Office 97-et támadó makróvírusok. Az első ilyenek a Word 6/7/95 makróvírusainak automatikus lefordításával keletkeztek, de gyorsan feltűntek a kifejezetten Office 97 dokumentumokra írt vírusok.

Márciusban jelent meg a WM.ShareFun nevű Word 6/7 dokumentumokat fertőző makróvírus, mely amellett, hogy a Word szokásos eszközeinek felhasználásával szaporodott, további terjedési lehetőségként az MS-Mail segítségével szétküldte a fertőzött dokumentumok másolatait.

1997 áprilisában találták meg az első olyan hálózati férget, mely saját kódjának terjesztésére az FTP (File Transfer Protocol) protokollt használta fel.

Júniusban megjelent az első önkódoló Windows 95 vírus.

Novemberben bukkant fel az Esperanto vírus, mely a DOS, és a Windows32 végrehajtható fájljai mellett a Mac OS (Macintosh) fájljait is képes volt megfertőzni. Szerencsére a vírus a benne maradt programozási hibák miatt mégsem volt képes a multiplatformos terjedésre.

Decemberben egy új vírustípus jelent meg, az ún. mIRC féreg. Az egyik legnépszerűbb Internetes csevegő (Internet Relay Chat, IRC) eszköz, a mIRC sérülékenységét használta ki, mely lehetővé tette, hogy vírus scriptek továbbítsák kódjukat az IRC csatornák között. A következő IRC kiadás már blokkolta ezt a rést és ezek a férgek lassan eltűntek.

1998 eseményei

Továbbra is folytatódtak a Windows, a Microsoft Office és a hálózati alkalmazások elleni támadások. A számos trójai program mellett többféle rejtőzködő rendszeradminisztrációs program is feltűnt a számítógépes világban. Továbbra is előfordultak vírusfertőzött CD lemezek a magazinok mellékletei között (főleg CIH és Marburg vírusok).

Az év a W32/HLLP.DeTroie víruscsalád megjelenésével kezdődött, amely nem csupán a végrehajtott Windows32 programokat fertőzi meg, de emellett képes továbbítani a vírus készítőjének a megfertőzött számítógép információit. A vírus speciális könyvtárakat használt, melyeket csak a Windows francia kiadásában találni meg, ezért a fertőzés csak a francia nyelvű kiadásokat érintette.

Februárban az Excel táblázatokat fertőző vírusok egy új típusa jelent meg. Ez az Excel4.Paix (más néven Formula.Paix, XF/Paix) volt. Miközben beépült az Excel számolótáblákba, a víruskód tárolására nem a szokásos makróterületet használta, hanem a formulák területét.

1998 februárjában és márciusában detektálták a Win95.HPS és Win95.Marburg vírusokat, amelyek az első "élő" polimorf Windows32 vírusok voltak.

1998 márciusában jelent meg AM.AccessIV néven az első Microsoft Access makróvírus. E vírusnál nem volt hasonló felfutás, mint a többi makróvírusnál, mivel meglehetősen kevesen csereberélnek Access adatbázisokat. Ugyancsak ebben a hónapban jelent meg a OM97.Cross makróvírus, amely már két különböző Office program, az Access és a Word dokumentumait volt képes megfertőzni. Ezek után még néhány további vírus is előkerült, amelyek kódjaikat két vagy több Office program között is át tudták vinni.

1998 májusában jelent meg a Windows EXE fájlokat fertőző és a fertőzött fájlokat az Eudora e-mail levelező program segítségével is továbbterjesztő RedTeam vírus első, majd hamarosan módosított második kiadása is.

Az év közepén, 1998 júniusában a Win95/CIH vírus okozott pusztító járványt. A Tajvanról származó vírus megpróbálta, a Pentium processzoros PC-k flash-EPROM alapú BIOS-át felülírni, törölni. Ez a tönkretett flash-EPROM cseréjéig vagy újraprogramozásáig használhatatlanná teszi a számítógép alaplapját, s ezáltal az egész számítógép működésképtelenné vált.

Augusztusban jelent meg a BackOrifice nevű számítógép és hálózat távmenedzselésére alkalmas segédprogram. Elsődlegesen rosszindulatú támadók használták fel a rendszerek

fölötti irányítás átvételére. Ez a program a trójaiak új generációjának első tagja. Ezt követően hozzá hasonló programok sora jelent meg, NetBus, Phase, Sub7, stb. Ugyancsak augusztusi esemény az első Java alapú végrehajtható programokat fertőző vírus, a Java/StrangeBrew megjelenése. Ez a vírus egyelőre nem jelent veszélyt az Internet felhasználóira, mivel nincs arra mód, hogy a vírus szaporodási funkciói bármely távoli számítógépen működjenek. Azonban a vírus megjelenése rávilágított arra, hogy az Interneten szörföző gépeket is érheti vírustámadás kliens programokon keresztül.

Novemberében megjelent a VBS.Rabbit nevű víruscsalád, mely a Webes fejlesztésekben kiterjedten használt VisualBasic scripteket alkalmazta. E vírusok után természetszerűleg következett a VBS.HTML.1nternal felbukkanása, amely egy HTML vírus. A víruskészítők figyelmüket a hálózati alkalmazások támadása felé fordították, új hálózati férgek, vírusokat fejlesztése történik, melyek a Microsoft Windows és az Office programok tulajdonságait használják fel a távoli számítógépek és webszerverek megfertőzésére, valamint tömeges küldés révén terjesztik saját kódjukat az elektronikus levelező rendszereket felhasználva.

Végül pedig decemberben megjelent az első PowerPoint formátumot támadó vírus is, az Attach, melyet rövidesen követett a ShapeShift és a ShapeMaster. Ennek a formátumnak a vbs moduljai tömörített módon tárolódnak, melynek átvizsgálása komplexebb feladat, azonban a legtöbb védelmi szoftvergyártó cég sikeresen integrálta ezt szoftvereibe.

1999 eseményei

Az év első kiemelkedő eseménye a március 26-án megjelent WM97/Melissa makróvírus. A Melissa volt az első olyan makróvírus, amely levelező rendszereket használt fel a víruskód terjesztésére, méghozzá igen hatékony módon. Napok alatt az egész világon elterjedt és több milliónyi számítógépet fertőzött meg és a sok önmaga terjesztésére szánt levéláradat végül szerverek százait terhelte túl. A Melissa után, mindössze pár héttel később megjelent a XM97/Papa nevű levelező Excel makróvírus. Nem sok különbség volt a terjedési algoritmusban, a Melissa első változata 50 címre küldte el magát, a Papa makróvírus pedig 60 helyre postázta el a víruskódját hordozó fertőzött leveleket.

A két nevezett makróvírus megjelenése után százával jelentek meg a Melissa és Papa átiratok, majd újabb, ezekkel már nem rokon makróvírusok.

A fejlődés a scriptvírusok területén sem állt meg. A VBScript és JavaScript vírusok hihetetlenül nagy mennyiségben bukkantak fel. A script alapú kártevők közül egyedülálló tulajdonságot azonban csak kevés mutatott. A VBS/BubbleBoy azonban egészen kiemelkedett

innen újszerű módszerével, mely az Outlook egyik biztonsági rését kihasználva fertőzött. A víruskódja nem csatolt fájlban, hanem magában a levéltörzsben van, és a Preview (előzetes megtekintés) ablak hibáját kihasználva a fertőzött levél első néhány sorának megjelenítésekor is fertőzött. A HTML formátumú levelekben elhelyezett script kódot ugyanis már ilyenkor is végrehajtja az Outlook.

A Scriptvírusok családja 1999-től már nem csupán a VBScript vírusokból áll, ezen kívül tartalmaz JavaScript és CorelScript kártevőket is. A Windows32 férgek közül a Happy99 jelentősebb ebben az évben.

Elterjedt a ZippedFiles nevű internetes féreg, mely különböző alkalmazások fájljait törölte, működésképtelenné téve őket.

Novemberben megjelent a BubbleBoy, majd rövidesen utána a KakWorm, melyek az első olyan férgek közé tartoznak, melyek emailen keresztül, de csatolt fájlok nélkül terjedtek. Ezek a programok az Internet Explorer hibáját használták ki.

A Babylonia-val megjelent ez első olyan féreg, mely képes volt önmagát frissíteni.

2000 eseményei

2000-ben folytatódott a scriptvírusok térnyerése. A fertőzési statisztikák szerint a makróvírusok már ebben az évben visszaszorultak a második helyre. A harmadik helyet pedig a Windows32 vírusok és férgek foglalták el. A hagyományos fájlfertőző és bootvírusok erősen visszaestek, a statisztika legvégére kerültek. Ebben az évben olyan férgek bukkantak fel, mint a W32/ExploreZip, a W32/Hybris, a W32/Navidad és a W32/Sircam. Ezek a kártevők szinte kizárólag elektronikus levelek útján kerültek a számítógépekre.

A script-kártevők közül a leghíresebb a VBS/LoveLetter lett. Továbbra is jelennek meg új makróvírusok, de a fertőzési statisztikákat immár a script-kártevők és a 32 bites levelező férgek vezetik. A weboldalakon keresztül támadó szkriptvírusok is elterjednek.

A vírusok egyik különlegesebb képviselője az AutoCad csomagok fertőzésére írt Star, mely első a maga nemében.

Különböző platformok és technológiák első vírusai jelennek meg: Star(AutoCad), a Liberty(Palm OS), Pirus(PHP), Fable(PIF fájlok), Stream(NTFS-ADS), valamint 37 új trójai és vírus jelent meg a Linux operációs rendszerre.

2001 eseményei

Elterjedtek a különböző sérülékenységeket kihasználó férgek. A W32/CodeRedAlert két változatban is bejárta a világot. Az IIS biztonsági rését kihasználó program számos Internet-szolgáltatónál és több ezer kisebb cégnél okozott gondokat. A Kínából származó féreg meglepően sok kárt okozott, de sokkal kevesebbet, mint amit a sajtójelentések alapján várni lehetett volna. Köszönhető ez annak, hogy a Windows rendszerek visszaszorultak az internetes szerverek között, és a Unix, valamint Linux alapú szerverek nyertek teret. Ahol nem IIS futott, azok a helyek védettek voltak a féreg támadásával szemben.

A fent említett CodeRedAlert tanulságait, tapasztalatait használta fel a W32/Nimda (Admin megfordítva!) készítője. A féreg egyike a legagresszívabb kártevőknek és hónapokig a fertőzési statisztikák élén állt, ez összes lehetséges terjedési módszert használva, nem csak levelezést.

Új technológiákat hasznosítanak a vírusírók: IM, P2P, valamint a Linuxot is egyre több támadás éri. A közhiedelem szerint, amit a vírusszakértők többsége is támogat, a Linux sokkal biztonságosabb, mint a Windows, legalábbis vírusvédelmi szempontokból (amennyiben megfelelően van beállítva!).

A Linux/Winux, illetve Lindose néven megjelenő kártevő az Esperanto után a második olyan programkártevő, amelyet mind a Windows, mind a Linux rendszerek megfertőzésére képes.

Megjelennek az első fájlnélküli, csak memóriába töltődő férgek.

Az év elején számos DoS (Denial of Service) támadást regisztráltak világszerte, természetesen elsősorban az USA-ban.

2002 eseményei

A 2002-es fertőzési statisztikák élén továbbra is a Windows32 férgek állnak, de mögöttük szoros a verseny a makróvírusok és a script-kártevők között. A tendenciákat jelzi, hogy emelkedik a Linux rendszereken futó, script-férgek száma. A fő célpont a Microsoft Outlook levelező ügyfélprogram és a Microsoft Internet Information Server (IIS), illetve az Internet Explorer. Sorozatban fedeznek fel biztonsági réseket, melyek folyamatosan támadási felületeket adnak a vírusok számára.

Az év legkiemelkedőbb kártevője a W32/Klez, amelynek változatai különféle egyéb vírusokat tartalmaznak, így például a pusztító CIH vírus variánsát is. Az első négy hónap fertőzési statisztikái szerint csak a W32/Sircam és a W32/BadTrans terjedt el gyorsabban és nagyobb

számban. A Scalper és a Slapper férgek megjelenése megmutatta, hogy a nem-Windows használók sincsenek biztonságban.

A Donut személyében megjelent az első .NET-re írt vírus is.

2003 eseményei

Elmondhatjuk, hogy ez az év az internet férgek éve volt, számos járványnak lehettünk tanúi ebben az évben.

A Slammer eljövetelével új pontjára érkeztünk a vírustörténelemnek. A fájl nélküli és "Warhol" technikát használó féreg mindössze egy UDP(376 byte) csomagon keresztül terjedt, s a legtöbb sérülékeny számítógép a terjedés első 10 percében már áldozatául eset. A féreg az MSSQL szerverekben található sérülékenységet használta ki.

A következő széles körben elterjedt féreg a Lovesan(Blaster) volt, mely az RPC DOM sérülékenységet használta ki, mely mind a Windows XP-ben, mind a Windows 2000-ben megtalálható volt. A megfertőzött gépek elosztott terheléses támadást indítottak kijelölt webcímek ellen. Még ebben az évben megjelent a Welchia féreg, mely a sérülékenység javítását telepítette a gépekre és letörölte a Lovesant.

Már januárban detektálták a Sobig család első tagját, melynek "f" változata a mai napig is a legelterjedtebb email férgek között van. Természetesen számos másik email féreg is elterjedt (Tanatos, Lentin, Mimail, Sober), valamint a Witty, egy ICQ sérülékenységet kihasználó fájlnélküli IM féreg.

Jelentősen megemelkedett a kémszoftverek száma is, ezek közül az Agobot és az Afcore a legjelentősebb. Ezenkívül megjelent a proxy trójai, mint új programfajta, a spammerek új küldési módszerhez jutottak.

Különböző felderítő férgek is megjelentek, melyek hálózati helyeket, erőforrásokat, fedeztek fel.

2004 eseményei

Ebben az évben is folytatódtak a különböző email férgek támadásai, viszont megjelentek az egymás által hátrahagyott lyukakat, sérülékenységet kihasználó férgek.

Januárban a MyDoom és a Bagle első variánsai kezdtek terjedni, majd februárban megjelent a Netsky első változata, s evvel megkezdődött a két utóbbi hosszantartó háborúskodása, melyre később lesz példa más kártevők között is.

Két további kiemelkedő programférget kell még megemlítenünk: a Sassert, valamint a Plexust, melyek közül az utóbbi azért volt jelentős, mivel minden lehető terjedési módot felhasznált (P2P, email, hálózati megosztás). Szerencsénkre, mivel nem használt tömeges levélküldést, és social engineering technikákat, nem okozott világméretű járványt.

A különböző kártékony kódok együttműködése fokozódik, a férgek trójaiak telepítését végzik, melyek később frissíthetik az adott vírust.

Új platformok, architektúrák kerülnek a vírusírók látókörébe: a PDA-k, mobiltelefonok, Symbian, PocketPc.

Egy új ötlet merült fel a vírusírók részéről: használjunk Google-t! A Santy féreg a kereső motor segítségével kereste meg a phpBB portál motorban a sérülékenységet, s átírta ezeken az oldalakon a kezdőlapot. A jelentősége a módszernek a sérülékenység keresésének újszerűsége. Így a támadott szerverek közelében sem járt a támadó, így a naplók se mutattak ki semmit.

2005 eseményei

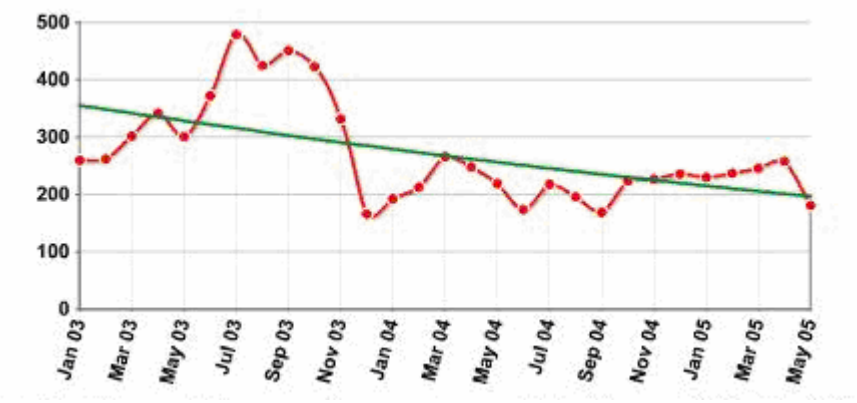
A tömeges levélküldő, IM, P2P férgek továbbra is terjednek (MyDoom, NetSky, Bagle, Mytob, Zafi, stb.), ezenkívül egyre több mobil kártevő jelenik meg, melyek főleg a Symbian operációs rendszert támadják, valamint megjelenik az első MMS-ben terjedő virus, a CommWarrior.

Az év legnagyobb járványa az MS05-039 sérülékenységét kihasználó férgeké (Ircbot, Zotob, Bozori), melyek már a sérülékenység kiadásától számított 4 napon belül jelentkeztek. Bár főleg Windows 2000-es operációs rendszeren működtek, így is nagy fennakadást okoztak. Ezek a férgek is versenyeztek egymással.

A rootkitek is egyre népszerűbbé váltak, immár Windows és Linux operációs rendszeren is nagyon sokféle van. Szinte versenyzés van a védelmi szoftverek és a rootkitek között. A Linux népszerűségét méltán jelzi, hogy több mint 700 kártevő van már az operációs rendszerre.

2.2. Tendenciák az elmúlt pár év alapján

Milyen tanulságokat lehet levonni az elmúlt időszak eseményeiből? Lássunk először is néhány tényadatot. 2003 januárja és 2005 májusa között eltelt időszak a következő számadatokkal szolgált:



1. ábra:A vírusok, férgek számának növekedése 2003-2005 között

Viselkedés típus	Növekedési arány a 2003-as évhez képest 2004-ben	Növekedési arány a 2004-as évhez képest 2005-ben
Email féreg	-20%	8%
IM féreg	Havonta 1 jelent meg	Havonta 8 jelent meg
IRC féreg	-28%	-1%
Hálózati féreg	21%	29%
P2P féreg	-50%	-36%
Féreg	-1%	24%
Vírus	-54%	-28%
Vírus és féreg összesen	-37%	7%

2. ábra : A vírusok, férgek számának változása típus szerint

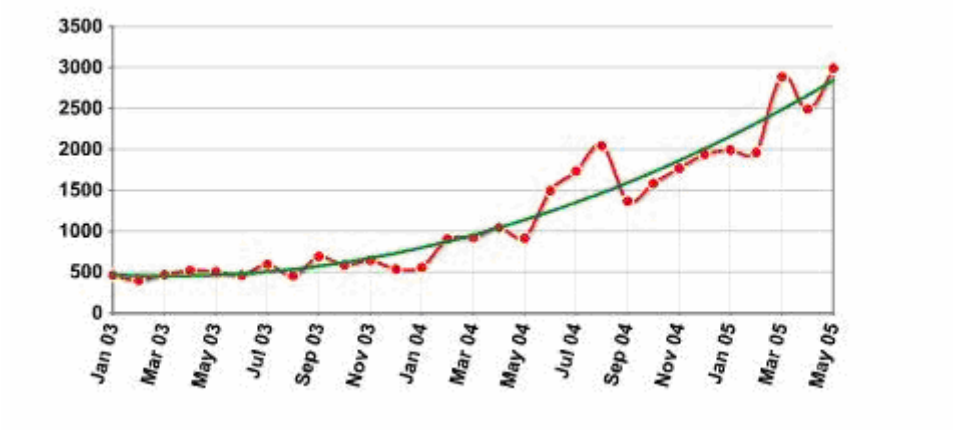
Jól látható a 2003-as év végén bekövetkezett nagymértékű csökkenés, azonban ez elsősorban az érdeklődés megváltozásának szól. A táblázatban (2. ábra) jól látszik a részletes változás. Az email férgek esetén a 2004-es 20%-os csökkenést kompenzálja egy 8%-os növekedés a 2005-ös év elején. Kérdés, hogy ez valódi növekedést fog-e jelenteni az év további részében. Valószínűleg nem fogunk látni újabb látványos járványt, részint a felhasználói tudatosság kialakulása miatt, valamint olyan új technológiák alkalmazása miatt, mint például a jelszóval védett zip fájlok belső, vagy a beérkező csatolt állományokat tartalmazó levelek előzetes vizsgálata.

Az IM férgek 2005-ben stabil növekedést mutatnak, mely nem véletlen, részint újdonságnak számítanak, másfelől nincs meg az a tudatosság a felhasználók részéről, mint például a beérkező elektronikus levelek esetén. Bár maga a protokoll képes lenne fájlátvitelre is, azonban valószínűleg a programozók képzetlensége miatt, inkább a linkekre kattintás révén töltődnek le a célgépre. Egyelőre hasonló utat járnak be, mint az email férgek 2002 és 2004 között.

Az IRC férgek lényegében eltűntek, legtöbbjük trójaiként jelenik meg újra a színen.

A hálózati férgek folyamatos fejlődést és növekedést mutatnak, sikerességük egyik titka, hogy a terjedéshez semmiféle felhasználói interakció nem szükséges. Jelenleg kezdik átvenni az email férgek helyét. Mivel általában tartalmaznak egy trójai komponenst is, a botnetek kiépítése is könnyedén megtörténhet.

A P2P férgek hasonló szintre estek vissza, mint az IRC kártevők. Ennek egyik oka lehet a médiatársaságok erőteljes kampánya a technológia ellen. A klasszikus vírusok majdnem teljesen eltűntek, mely főleg a lassú terjedésnek tudható be. Bár nem kérdéses, hogy a lassú terjedés a detektálást is nehezebbé teszi, hiszen nem okoz akkora anomáliákat.



3.ábra: A trójai programok számának növekedése 2003-2005 között

Viselkedés típus	Növekedési arány a 2003-as évhez képest 2004-ben	Növekedési arány a 2004-as évhez képest 2005-ben
Bank	170%	115%
Hátsó kapu	41%	49%
Trójai	537%	74%
Trójai-ArcBomb	-	-
Trójai-Clicker	283%	83%
Trójai-DDoS	-	-
Trójai-letöltő	-	184%
Trójai-telepítő	-	61%
Trójai-IM	-	-
Trójai-értesítő	-	-
Trójai-proxy	-	61%
Trójai-jelszó szerző	52%	47%
Trójai-kém	251%	71%
Rootkit	Átlagosan 6,25 jelenik meg havonta	Átlagosan 17,8 jelenik meg havonta
Trójai összesen	157%	82%

4. ábra: A trójai programok számának változása típus szerint

A trójai programok ez alatt az időszak alatt a vírusírók kedvenc fegyvereivé váltak. Minden fajta típusuk növekszik, s a downloader/dropper típusai jól használható botnetek létrehozására. Nem véletlen, hogy ez a két alfaj a legdinamikusabban növekvő trójai programtípus. Napjainkra az is jellemző, hogy bérbe adják egymást közt az így kiépített botneteket.

A jelszó, és egyéb adatokat lopó programok is szignifikáns növekedésnek indultak, nem véletlenül: a vírusírókat erősen motiválja a haszonszerzés, s ezekre van kereslet az interneten, vagy éppen maguk használhatják fel bankkártya-csalások, identitáslopások, és egyéb pénzszerzési akciókban.

A rootkitek különösen népszerűvé váltak, amely avval magyarázható, hogy hosszútávon talán ezeknek a programoknak van a legnagyobb túlélési és működési esélyük, vagy segíthetik a már telepített kártékony programokat, megóvva őket. Windows rendszeren elsősorban a rossz felhasználói beidegződésnek köszönhetően képesek települni, hiszen a legtöbben rendszergazdai jogosultságokkal vannak bejelentkezve, s innentől kezdve a program bármire képes lehet (Bár korlátozott felhasználóként nehézkes lehet a Windows használata).

Megfigyelhető, hogy a vírusírók új platformok felé fordulnak, elsődleges célpontjaikká váltak: Unix, mobil, .NET.

Év	A megjelenő kártékony programok átlagos száma havonta
2003	12,67
2004	21,58
2005	35,20

5. ábra: A Unix rendszert fenyegető kártékony programok számának növekedése

Lassan kezd eloszlani az a tévhit, hogy a Linux nem sérülékeny. Sőt, ha jobban megnézzük a különböző detektált sérülékenységek listáját, gyakran láthatjuk, hogy bizony jóval több sérülékenységet tartalmaz, mint a Windows. Természetesen ez a különböző menedzselési politikából is adódik…

Napjainkban már naponta szükséges lehet frissíteni a szoftvereket.

A mobil eszközökben elsősorban Symbiant (mobiltelefonok), WinCE-t (Pocket PC-t) (PDA-k), Linuxot és Palm OS-t használnak, bár az utóbbi kettő nem kimondottan elterjedt. Ebből következően a vírusírók is elsősorban az első kettőt támadják. A Symbianra 2004-ben jelent meg az első koncepció vírus, s azóta folyamatosan egyre több jelenik meg. Eleinte trójaiak

voltak, majd hamarosan megjelentek a fájlfertőző, majd a bootolást is irányító vírusok, néhány képes volt smst is küldeni. Elsősorban bluetooth és MMS a terjedési módszer, fontos azonban megjegyezni, hogy idáig egy sem volt, mely valamilyen rendszer sérülékenységet használt volna ki.

A WinCE platformra még nem készült túl sok vírus, bár folyamatosan jönnek ki újabb és újabb példányok.

[3], [5], [6], [23]

2.3. Változás a hálózati viselkedésben

Az elmúlt évben több igen súlyos biztonsági incidensnek lehettünk tanúi. Jól látható, hogy a támadási vektorok megváltoztak. Míg régebben DOS, DDOS tevékenység által okozott károk voltak a jellemzők, addig jelenleg a több pénzügyi vállalat ellen indított támadások okoztak nagy károkat. A Bank of America, Sumitomo Bank, Master Card és a Visa estek cyberbűnözök prédájául, de a Hotworld incidens sem elhanyagolható jelentőségű, melynek során több mint 80 izraeli és egyesült királyságbeli szervezet hálózatában találtak trójai kémprogramot.

Nem véletlen, hogy a támadók ebbe az irányba fordultak, hiszen az antivírus iparnak már legalább évtizedes tapasztalata van a széles skálájú támadások területén, és ki is fejlesztette a megfelelő megoldásokat. Ez a legegyszerűbb megoldástól – ugyanazon levél többszöri detektálásától – komplexebb megközelítésekig, mint tűzfal és IDS-IPS tart. Ma 1 óránál kevesebb is lehet a reakcióidő, hogy az adott férget detektálják és felvegyék a vírusadatbázisba. Másodsorban, még ha egy féreg át is tör a sokszoros védelmen, és megfertőzi pár ezer belső felhasználó gépét, tovább kell terjednie és legfőképpen továbbítania önmagát. Mivel a kártékony program visszafejtésével meg lehet tudni a szervert, illetve a csatornát mellyel a támadó irányítja programját, hálózatát, a szerver lekapcsolásával működésképtelenné tehető az irányítás és az adatátvitel. Harmadrészt a megszerzett adatokat fel is kell használni, s a mai világban ez már nem triviális feladat.

Maga a betörési módszer jelentősen különbözik a már megszokottól. Egy dolog megfertőzni egymillió számítógépet, s róluk 50 ezer hitelkártya számot megszerezni, s megint más dolog 1 számítógépről megszerezni egymillió kártyaszámot. A CardSystem Solutions incidens elemzése ad néhány támpontot. Először is az a kártékony program, mely megfertőzte a rendszert, még mindig nem jutott el az antivírus cégekhez. Ezzel ellentétben a HotWorld

incidensben szereplő trójai 2 nappal később már megtalálható volt a vírusadatbázisokban. Másodszor, maga a trójai nem lehetett valamilyen egyszerű billentyűzetleütést naplózó program, hiszen valószínűtlen hogy azt a 40 millió kártyaszámot manuálisan gépelték be. Ebből következik, hogy a programnak az adatbázishoz kellett hozzáférnie, s ennek érdekében a CardSystem rendszerére kellett speciálisan készíteni. A rendszer határain belül való információtovábbítás továbbra is kérdéses.

Több incidens is visszhangot váltott ki a médiában, s ezek alapján állíthatjuk, ezek a bűnözök már készek kifizetni több tízezer dollárt, hogy belső információkat szerezzenek, vannak belső kapcsolataik a szervezetben, s tudják, hogyan kerüljék el az IDS-eket, védelmi szoftvereket. Végezetül a megszerzett információkat nem nyíltan (még csak földalatti fórumokon sem) értékesítik. Miért hallunk egyre több és több esetben hasonló támadásokról? Minél nagyobb egy szervezet, annál sebezhetőbb az infrastruktúrája: sok számítógép, hibrid hálózatok, tele különféle hozzáférési jogosultságokkal, sok dolgozóval, mind-mind megkönnyítik egy támadás megkonstruálását. Néhány nagyobb hálózatban egy dokumentum megtalálása is lehetetlen feladat. Hogyan vegyünk észre egy trójai programot? Ráadásul egy ilyen trójai egyedi, gyakran csak 1 példány létezik belőle, melyet igen speciális céllal írtak meg. Ez azt jelenti, hogy szinte lehetetlen heurisztikus detektálást végrehajtani, s az antivírus cégek még csak nem is hallanak róla…

A program bejuttatása a rendszerbe továbbra sem egyszerű feladat. Az adott szervezet email címeire küldött rosszindulatú kódot tartalmazó levél nem jó megoldás, hiszen evvel a tűzfallal, víruskeresőkkel nem védett rendszereket lehet csak bevenni. Gyakoribb megoldás a böngészők hibáinak kihasználása, s a felhasználó átcsábítása egy adott oldalra, mely tartalmazza a telepíteni kívánt kódot. Ennek módszere lehet valamilyen spam email, vagy IM üzenet küldése. Azonban ezek az oldalak nem túl hosszú életűek, a különböző biztonsági cégek gyakorta kapcsoltatják ki/töröltetik le őket. A másik megoldás egy ismert, és sűrűn használt oldal feltörése, s a kód ebbe az oldalba történő beillesztése. Legnépszerűbbek a php motorral működő különböző CMS megoldások, melyekben folyamatosan fedeznek fel hibákat. Erre a megoldásra komoly példa is volt ebben az évben, az MSN Korea oldalát törték fel a nyáron, s 5 napig az odalátogató felhasználók gépeire letöltődött a trójai program.

Ebből a szempontból nem megvetendő célpontok a mostanában oly népszerű online multiplayer játékok sem…

Az elmúlt 3-4 évben a klasszikus vírusok szinte teljesen eltűntek. Nem volt olyan, mely féregtulajdonságok nélkül képes lett volna egy nagyobb globális járványt okozni. Egy olyan

fájl vírust létrehozni, mely több Windows verzióban is működőképes, komoly programozói tudást igényel. Ráadásul a polimorfizmus fontos tulajdonsága kell, hogy legyen ezeknek a programoknak, ami annyit jelent, hogy a különböző kódolási algoritmusokhoz is kiválóan értenie kell az alkotónak. Bár ez alatt az időszak alatt nem találtak jelentősebb vírusokat, azonban mégis jelen maradtak. Hogy miért nem használnak mégsem ilyen vírusokat? Mert az antivírus cégek rákényszerítették a vírusírókat, hogy programjaik új módon fertőzzenek, és rejtsék el jelenlétüket, melynek legnépszerűbb módja a rootkitek használata, vagy rendszerfájlokba történő kódinjektálás. Az év során detektáltak olyan trójait, mely legális fájlnak, alkalmazásnak álcázták magukat, s EPO technikát használva kerültek bele a programba (például winrar.exe). Szerencsére ezek a programok valamilyen ismert botnet létrehozó program kódját használták, így a detektálásuk az antivírus szoftverek hatókörén belül voltak (Agobot, Rbot or SdBot). Egy másik program a rendszerkönyvtárat fertőzte meg (wininet.dll), s minden hívást elfogott. Amennyiben a program egy legális alkalmazáshoz kapcsolódik s úgy kommunikál a külvilággal, különösen veszélyes lehet. Szükségessé válik hatásos fájlszkennelő programok fejlesztése, mely képes a fájlok tartalmának változását is lemérni, főleg abban a viszonylatban, hogy esetleg a fájl mérete nem is változott meg.

2.4. Mire számíthatunk a jövőben?

Regionális spam programok, melyek alkalmazkodnak a helyi nyelvhez, nehezebbé téve detektálásukat, s megkülönböztethetőségüket az igazi levelektől.

Célzott levélküldések, vagy egyéb penetráció, melynek során a kártékony programokat 1-1 kiválasztott szervezethez juttatják el, így az antivírus cégek is nehezebben tudnak reagálni rájuk (ha eljut hozzájuk egyáltalán 1 példány belőle). Jellemzőjük lesz a lopakodó tulajdonság, ha kicsit szélesebb körben támadnak, valamilyen polimorf motor.

Csökkenő számú globális féregjárványok, melyek továbbra is valamilyen frissen fellelt sebezhetőségen alapszanak majd.

Új „social engineering” technikákat használó programok fejlesztése. (PSW.Win32.LdPinch – ICQ, mely még beszélget is a felhasználókkal).

A szervezett bűnözés bekapcsolódása az Internetes támadásokba, esetleges háborúik is ezen a fronton zajlhatnak le.

A vírusírók új platformokat támadnak meg, melyek népszerűvé válnak, s sok felhasználót vonzanak.

A növekvő funkcionalitású mobil eszközök nagyobb fokú információtárolást biztosítanak, így ezek növekvő érdeklődésre tarthatnak számot a rosszindulatú felhasználók részéről.

Felvetődik a kérdés az olvasóban, hogyha a férgek, vírusok, trójaiak ennyire visszaszorulóban vannak, akkor voltaképpen miről beszélünk? Érdemes-e erre iparágat építeni, vagy hamarosan megoldódnak ilyen jellegű problémáink? A következő fejezetekben kielemzem a lehetőségeket. Elöljáróban viszont meg kell határozni, hogy a vírusfajok alatt mit is értünk. Sokféle definíciót találhatunk, melyek közül véleményem szerint ez a három a legjobb a fenti 3 kártékony kód típus definiálására:

Férgek:
Olyan programok, melyek önmagukban is működőképesek, nem szükséges valamilyen programhoz kapcsolódniuk, reprodukcióra képesek és terjesztik magukat.

Vírusok:
Olyan programok melyek valamilyen hoszt programhoz kapcsolódnak, s futásuk ezen programoknak a futásához kapcsolódik. Szintén képesek önmaguk reprodukálására és terjesztésére.

Trójai programok:
Olyan program, mely nem képes önmaga reprodukálására és másolására, viszont a felhasználó tudta nélkül enged hozzáférést a számítógéphez, végezhetőek el vele valamilyen műveletek. Gyakran a felhasználó megtévesztésével kerül fel a gépre. Általában a felhasználó megtévesztésén alapszik működésük.

A klasszikus definíciók közötti határok kezdenek elmosódni. Hiszen most már együtt jár egy féreg terjedésével, hogy működése során trójait telepít a rendszerre, vagy a trójai fájlokba épül be, s ez tipikus vírusviselkedés, attól eltekintve, hogy nem többszörözi magát, s nem terjed tovább. Bár a jövőben az is elképzelhető, hogy a kiválasztott céloknak megfelelően mégis továbbterjed 1-1 hosztra. Véleményem szerint a jövő hírszerző szoftverei mindhárom tulajdonságot egyesíteni fogják magukban.

[13]

3. Technológiai lehetőségek

Az elmúlt részben láthattuk, milyen fejlődésen mentek keresztül a kártékony kódok. Kísérletekből, elrontott programokból egész komoly gazdaságot hátráltató tényezők lettek. Fontos megértenünk, hogy milyen védelmeket kell kidolgoznunk a közeljövőben ellenük. Ebben a szekcióban áttekintést fogok adni a jelenleg használatos hálózati, memória, fájl és hardveres lehetőségekről, illetve a várható fejlődési irányzatokról. Ezt követően a lopakodó tulajdonság lehetőségeire térek ki, melynek jelenleg két leggyakrabban használt eszköze a rootkit technológia és a kriptográfia.

3.1. Fájl technikák

Felülírás:

A fájl egy részét felülírja program, s elhelyezi a saját kódját benne. Ez a legkönnyebben megvalósítható, ugyanakkor a legkönnyebben detektálható lehetőség. Többnyire a felülírt alkalmazás nem fog működni, ráadásul a fájl megváltozása könnyen kimutatható. Azonban adatpusztításra kiválóan alkalmazható és egy ilyen triviális program mérete nagyon kicsi lehet. Lehetséges a BIOS diszk műveleteit felhasználni az operációs rendszer írási/olvasási műveletei helyett. Véletlen algoritmust használva a fájl egy találomra kiválasztott részét írja felül. Semmilyen esetben nem állítható vissza a fájl.

Hozzáfűzés:

Ebben az esetben a fájl elejéhez vagy a végéhez fűzi a kódját hozzá a program, s elhelyez egy ugrási utasítást a fájl elején. Indításkor az ugrási utasítás révén lefut a vírus törzse, majd visszaadja a vezérlést az eredeti programnak. Mivel ezt nem olyan könnyű feladat megoldani, mivel ügyelni kell, hogy a két program kódja ne kavarodjon össze, gyakran egy külön fájlba mentik más néven az eredeti programot, a vírus pedig megmarad az eredeti nevén. Ezután a vírus végrehajtódása után lefuttatódik az eredeti is.

„Cavity":

Ebben az esetben a programok a fájlok azon részét írják felül, amelyben nem tárolódik értékes adat. Például, ilyenek lehetnek a bináris fájlok 0-kat tartalmazó részei, vagy a space, vagy a c fordítók által használt 0xCC-vel töltött blokkok. Természetesen ez a fajta fertőzési mód már nagyban függ az adott operációsrendszertől. Máshogy lehet megfertőzni egy Windows PE, vagy egy Linux ELF fájlt, hiszen a blokkok felülírásának lehetősége a különböző fájlformátumok felépítéséből adódik (szegmensek, fejléc, belépési pontok, stb.). Mivel ez a

fajta fertőzési mód nem változtatja meg a fájl méretét, az egyszerű felhasználónak nem is tűnik fel.

„EPO”:

Ebben az esetben a fertőzés során nem változik meg a fájl belépési pontja, sem az ott lévő kód. A program kódja úgy épül be a fájlba, hogy véletlenszerűen fog lefutni, elhelyezve egy hívásutasítást. Windows környezetben ez technika meglehetősen hatásos lehet, hiszen a különböző API függvények híváspontjait magukra átirányítva képesek lefutni. Természetesen az alkalmazásból való kilépés átirányítása is megvalósítható, az ExitProcess() hívásának átírásával, amely a vírustörzsére mutat. További módszer, hogy a fájl import címtábláját írják át, így minden API hívás esetén először a vírus kódja fog lefutni. Jó védelmi módszer lehet az „ismeretlen” belépési pontok használata is. Például a Windows PE fájl esetén a fő belépési pont a PE.OptionalHeader.AddressOfEntryPoint mezőjében tárolódik. Ahova ez mutat, ott fogja a program futását indítani. Kevesen tudják azonban, hogy ez nem feltétlenül az első olyan belépési pont, melyet a rendszertöltő végrehajt. A Windows NT és az utána következő rendszerekben a PE fájl fejlécében lévő TLS adatkönyvtárat nézi és hajtja végre először. Egy kimondottan szofisztikus módszer a kódintegrálás, melynek során a vírus a hoszt program utasításai közé illeszti bele s sajátjait. Ez persze nagyon komplikált folyamat, mely során az eredeti program teljes visszafejtése is megtörténik, majd immár a víruskóddal való összeillesztése is. Ez eléggé processzor és memória igényes feladat. Ezeknek a vírusoknak frissíteniük kell az eredeti program adat és kódszekcióinak relokációs kapcsolatait. Nehéz egy ilyet megírni a feladat komplex volta, valamint a különböző alkalmazás, operációs rendszer verziók miatt. Azonban ha egy ilyen programot egy bizonyos operációs rendszer célalkalmazására írunk, nem lehetetlen, hogy kifizetődő lesz a dolog…

Végül pedig ennek egy lehetséges jövőbeli fejlesztése a kódépítés, mely során a hoszt fájl tartalmát használja fel a vírus kódjának felépítéséhez. Egy vírus képes lehet arra, hogy analizálja az eredeti program kódját olyan módon, hogy abból felépítheti saját utasításait. Képzeljük el azt az esetet, amikor megtalálta a megfelelő sztring darabokat a fájlban, s azt a memóriában rakja össze önmaga számára. Maga a kódépítő egy generikus kódsorozatnak látszódna, amely különböző metamorf technikákat használva meglehetősen változatos lehetne, valamint integrálódhatna magába a hoszt fájlba. Egy ilyen vírust detektálni nagy kihívás lenne a jelenlegi vírusvédelmi megoldások számára.

[1], [3]

3.2. Boot és egyéb technikák:

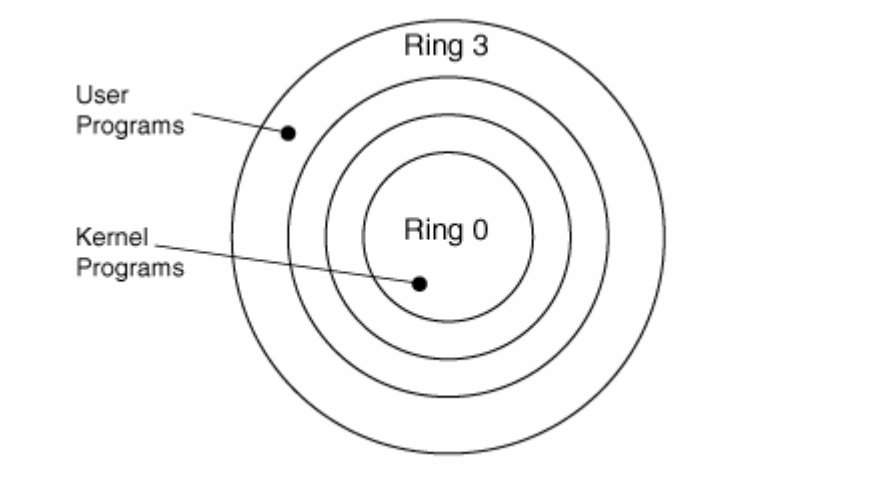
Idesorolom azokat a technikákat, melyek a merevlemez, s annak fájlrendszerét manipulálják. Az operációs rendszer indításakor betöltődő vírusok a partíciók, vagy az MBR adatait írták át. Az itt elhelyezett kód minden indításkor végrehajtódik, s egyből memóriarezidensé válhat. Így megvan annak a veszélye, hogy egy rosszul sikerült helyreállításnál a partíciós tábla, s így a merevlemez tartalma is elveszik. Ez persze azon is múlik, hogy a módosításkor elmentették-e az aktuálisan a partíciós táblában szereplő beállításokat, adatokat.

Esetenként a vírus kijelölt egy szektort a lemezről, s azt rossznak címkézve tárolta ott adatait, saját kódját, stb. Néhány esetben merevlemez utolsó szektorát használja, abban reménykedve, hogy az operációs rendszer nem fogja ezt a részt felhasználni. A partíciót kisebbre írva pedig, a vírus biztos lehet abban, hogy semmilyen más program nem fogja használni ezt a területet. Ezt egyébként most is ki lehet használni, hiszen telepítésnél az operációs rendszer lefoglal egy kis területet a telepítő fájljai számára, amiket később töröl, de a hely ott marad. Bár ezek a technikák manapság már nem népszerűek, lehet hogy hamarosan újra azzá válnak. A 2005-ös Black Hat konferencián bemutattak olyan technikát mellyel a tetszőleges bootszektor kódot felhasználva meg lehet változtatni a Windows kernelt, ahogy az az indulásnál betöltődik, rootkitet injektálva bele...

[1], [24]

3.3. Memória technikák

A mai processzorok mikrochipjei a hozzáférési privilégiumokat gyűrűkre osztva határozzák meg (Ring0-3). A Ring Zero a legmagasabb privilégiummal rendelkező. A Windows kernel kódok például kivétel nélkül a Ring0-ban, a felhasználói módú programok, pedig Ring3-ban futnak (általában a köztes szinteket nem használják). A CPU felelős azért, hogy kövesse, melyik program melyik gyűrűbe tartozik. A gyűrűk közötti átmenet pedig pontosan szabályozott. Csak nagyon ritka esetben engedett meg, hogy egy Ring3-ban lévő program hozzáférjen valamilyen kernel beállításhoz, esetleg egyes adminisztrátori programok rendelkeznek ilyen jogosultságokkal. Alapesetben az operációs rendszer nem engedi a hozzáférést, s ráadásul még megszakítás is generálódik ilyen esetben.



6. ábra: Privilégiumok elkülönülése

A CPU a Global Descriptor Table (GDT), Local Descriptor Table (LDT), Page Directory, Interrupt Vector Table (IVT) táblákkal tartja nyilván a jogosultságokat. Ezen kívül léteznek operációs rendszer specifikus táblák is, melyek azonban közvetlenül nem támogatottak a CPU által. Így tehát erős különbségek vannak felhasználói és kernel módú programok lehetőségei között.

Manapság a legelterjedtebb viselkedésforma a kártékony kódok esetén az, hogy miután valamilyen formában sikerült bejutniuk a rendszerbe, és ott lefutniuk (sérülékenység, felhasználói interakció, stb.) a memóriában maradnak, s rezidensé válnak. Ennek előnye, hogy az egész rendszer futását tudják befolyásolni, valamint védelmi intézkedéseket hozhatnak a különböző védelmi szoftverek ellen, például leállítva a futó folyamataikat, félrevezető adatokat juttatva el hozzájuk, vagy csak a futásukhoz szükséges fájlokat megtámadva, törölve, esetleg módosítva az adatbázisukat. Kétségtelen, hogy ennek megvannak a maga veszélyei is, hiszen egyes esetekben a detektálást is könnyebbé teszi 1-1 rosszul elhelyezett memória lap, vagy a túlzott CPU és memória használat, vagy az, hogy ugyanazon adat többféle lekérdezése más-más eredményt ad. Lássuk, milyen technikái vannak ennek (Ezekben a példákban a Windows operációs rendszer sajátosságai szerepelnek).

Felhasználói módú lehetőségek:

A programnak be kell valahogy töltődnie. Ezt elérheti valamilyen sérülékenység (jellemzően puffer túlcsordulási hiba) révén, azonban célszerű kihasználni az operációs rendszer specialitásait (például egy dokumentálatlan API hívást). Különböző memória területekre kerülhet a program. Például, foglalhat saját területet, és különálló szálként folyamatként futhat. Persze egy különálló folyamatot a legegyszerűbb érzékelni. Ennél trükkösebb megoldás lehet, hogy a hoszt program memóriaterületére írja magát, s az eredeti program egy

külön szálaként fut. Elindíthatja magát szervizfolyamatként is. Miután betöltöttük a memóriába a programot célszerű kicsit rejtve hagyni, erre szolgál a következő három módszer:

- Az import címtábla (import address table – IAT) alapján be lehet avatkozni a program működésébe. Amennyiben az alkalmazás egy másik bináris fájlban lévő függvényt kíván használni, akkor ezt importálnia kell. Ezeket tartalmazza az IAT. Az import táblában keressük meg a figyelni kívánt API függvények címeit (GetProcAddress, CreateFileA, stb.), majd ezt kikeressük az alkalmazás „.idata" szekciójából (a PE fájlok fejléce tartalmazza ezt a helyet), s átírjuk, hogy a vírus címére mutasson. Miután a vírus elvégezte a kívánt műveleteket, továbbítja a hívást az esetlegesen módosított paraméterekkel az eredeti címre, melyet saját címtáblájába mentett el. Miért is jó ez? Vegyünk néhány példát:

  A fájlműveleteket figyelve a kártevő tetszés szerint fertőzheti meg a megnyitásra kerülő fájlokat, így ráadásul nem is művel semmi gyanúsat, a fájlok amúgy is módosulnak. Vagy figyeli a víruskereső futását, s ha olyan fájlt próbál olvasni, amely fertőzött, akkor annak egy régebbi, még normál változatát adja vissza. Ha azt érzékeli, hogy megpróbálják visszafejteni (debuggerben futtatva), akkor letilthatja a billentyűzetet, vagy újraindítja a gépet, stb.

- Az inline függvények használatával a program magát a hívni kívánt függvényt írja át, pontosabban a függvény elejének néhány bájtját, amit persze elment a vírus kódjában. Ennek következtében a függvény lefutása során egyből a kívánt víruskódra ugrik, majd vissza a függvény további részére, függetlenül attól, hogy az alkalmazás maga hogyan valósítja meg a címzést a függvényhívás során. Ráadásul ez a módszer akár kernel módban is használható.

- A dll fájlok használata újabb lehetőséget rejt magában, a dll injektálás lehetőségét. Erre maga a Windows operációs rendszer ad több lehetőséget.

  A Windows registry révén beállíthatjuk saját dll fájlunkat (*HKEY_LOCAL_MACHINE\Software\Microsoft\WindowsNT\CurrentVersion\Windows\AppInit_DL*) kulcsértékként, mely be fog töltődni az adott alkalmazás indulása során. Másrészt egy futó folyamat területére betölthetünk egy dll fájlt, amit persze az sosem igényelt (LoadLibrary, FreeLibrary függvények). Lehetőségünk van továbbá a rendszerfájlokat is megfertőzni (dll), s ezek betöltődésekor a kártékony program is

betöltődik. Azt azért tudnunk kell, hogy ezek a rendszerelemek jól védettek, s legrosszabb esetben is naplózódik a módosításuk. Windows esetén a WFP pedig még ennél is többet tesz: visszaállítja az eredeti fájlt. Persze ezt is meg lehet kerülni, például kikapcsolva a registryben (Windows 2000-ben a kulcsérték = 0xFFFFFF9D állításával).

Kernel módú lehetőségek:

A felhasználói módú változtatásokat könnyű érzékelni. A kernel mód viszont messze nem ad lehetőséget ilyen egyszerű életre. Elöljáróban meg kell jegyezni, hogy a kernel memóriához nem férhetnek hozzá a folyamatok (két kivétel: ha a folyamatnak valamilyen debug privilégiuma van, vagy pedig ún. call gate-t valósít meg). Ebből következően a kernelhez device driverként lehet hozzáférni. Sok helyen lehet elhelyezni ún. „kampókat", lássunk ezek közül néhányat:

- A megszakítási tábla (interrupt description table - IDT) a programozók munkáját könnyíti, mivel a különböző funkciók, szolgáltatások címeit nem kell belefordítani a programba, elég az INT x utasítást kiadni, mely az adott szolgáltatáshoz irányítja a vezérlést. Ezek a megszakítások egyaránt lehetnek hardver vagy szoftver által generáltak. Két fontos dolgot kell itt megemlíteni. Az egyik, hogy minden processzornak saját IDT-je van (ez multiprocesszoros rendszerekben fontos, hiszen mindegyiket kötni kell a programhoz). Másrészt itt nem működik az eredeti függvény hívása, adatok szűrése, és visszatérés algoritmusa, mivel az IDT kampó csak átmenő függvény, nem kapja vissza a vezérlést. Azonban képes arra, hogy azonosítson, vagy blokkoljon kéréseket adott szoftverek esetén (például HIPS, vagy tűzfalak esetén)
- Hasonló technikát használva a System Service Descriptor Table (SSDT) segítségével is létrehozhatunk kampókat. A Windows futatható programjai natív módon is elérhetik az operációs alrendszereket (OS/2, POSIX, WIN32). Ezeknek a natív rendszerszolgáltatásoknak a címeit a SSDT-ben találhatjuk meg. Ez a tábla indexelhető a rendszerhívások által, visszaadva a memória címét az adott függvénynek. A System Service Parameter Table (SSPT) pedig a függvények paramétereinek a számát tárolja. A két tábla adatait a KeServiceDescriptorTable kapcsolja össze. Ha egy folyamat meghívja a rendszerszolgáltatást, akkor az átadódik a kernel vezérlő számára, mely ellenőrzi a paraméterek számát, és az index alapján megkeresi a függvényt, s meghívja. Ha az SSDT-ben lévő függvénycímet átírjuk a saját programunk címére, akkor ismét megtörténhet az adatok szűrése, manipulálása.

A legújabb rendszerekben (Windows XP, 2003) ez a terület írásvédett, s írási próbálkozás esetén rendszerhibát okoz. Azonban lehetséges megváltoztatni a memória hozzáférési jogosultságokat a Memory Descriptor List (MDL) felhasználásával. Ennek részleteivel itt nem kívánok foglalkozni. Különböző makrókat használhatunk a megfelelő kampók létrehozására (SYSTEMSERVICE, SYSCALL_INDEX, mindkettő a ZW, illetve az NT típusú függvényeket manipulálja. A ZW függvények a device driverek és egyéb kernel modulok számára exportált függvények, az NT függvények az SSDT-ben megtalálható védett függvények).

- A minden eszközmeghajtóban megtalálható függvénytábla szintén jó hely egy kampó létrehozására. A meghajtó telepítése során létrehoz egy mutatótáblát, melyben a különböző I/O Request Packet (IRP) kéréseket kiszolgáló függvények címei találhatóak meg. Ezek az IRP-k felelősek az írás, olvasás, lekérdezés, stb. műveleteiért. Ezeket a címeket átírva a saját programra irányíthatjuk át a végrehajtást. Azonban az IVT-hez hasonlóan ezek a függvények sem kapják vissza a vezérlést, így adatszűrésre ez sem használható.
- Egészen eddig a pontig különböző kampók létrehozásának lehetőségeiről volt szó. Ezeknek a módszereknek azonban több hátránya is van. Például adott esetben könnyen észre lehet őket venni. Másrészt a memória egyes részeinek csak olvashatóvá tételével is kivédhetőek az ilyen manipulációk. Egy újabb megközelítés az ún. Direct Kernel Object Manipulation (DKOM). Ebben az esetben a kernel objektumait, struktúráit módosítjuk. Ez viszont nagyon függ az aktuális operációs rendszer verziójától. További hátránya a módszernek, hogy sokkal mélyebb ismeretek szükségesek a struktúráról, annak kapcsolatairól, felhasználásról, valamint, hogy csak a memóriában lévő részek módosíthatóak, példának okáért ezért nem használható fájlok elrejtésére (hiszen a memóriában nincsen tárolva az összes fájl). Ezáltal lehetőség van folyamatok, eszközmeghajtók, portok elrejtésére, privilégiumok megváltoztatására.

Idáig a Windows operációs rendszerben rejlő lehetőségekre mutattam rá, azonban ne higgyük, hogy ezek csak ebben a rendszerben léteznek. A Linux különböző verzióiban is léteznek hasonló helyek, talán még régebb óta is. Felhasználói módban elsősorban a különböző bináris fájlok cseréjére s az általuk szolgáltatott inkorrekt adatokra épülnek. Kernel mód esetén a betölthető kernel modulokra (LKM) alapszik a technika. Ebben az operációs rendszerben lehetőség van újraindítás nélkül is bővíteni a kernelt. Azonban nem ez az egyetlen lehetőség. A /dev/kmem a kernel memóriáról tárol pillanatképet, ennek módosítására is lehetőség van. A

kernel image fájl módosításával pedig azt biztosíthatjuk, hogy következő újraindításnál a már módosított kernel töltődjön be.

[1], [2], [8]

3.4. Hardveres technikák:

Egy érdekes lépcsőfok a már nem csak RAM memóriát használó kártékony kódok esete. Az olyan programok, melyek fájlban tárolódnak, viszonylag feltűnőek, és egy gyanakvó adminisztrátor bármikor elindíthatja a gépet egy tiszta forrásról és különböző eszközökkel átkutathatja a tartalmát. Ez majdnem biztosan kiszűri a különböző „rootkit" technikával rejtett fájlokat, ugyanakkor ott maradnak azok a fájlok amik nem így vannak rejtve, s kártékony kódrészleteket tartalmaznak. Új lehetőségként azonban feltűnt az EEPROM-ban, flash memóriában elmenthető program vagy adat. Például a videokártya memóriája kiválóan alkalmas a célra. Bár pillanatnyilag gondot okoz a videokártyából a központi memóriába való írás módszere, odafele már működik a dolog. Valószínűleg a videokártyát is fel lehetne használni utasítások végrehajtására, lévén nem más, mint egy specializált processzor. Mint említettem az előszóban is, a fejlődés nem áll meg...

Nézzünk egy kicsit a múltba. Találkoztunk olyan vírussal, mely képes volt a BIOS memóriáját módosítani. Napjainkban az alaplapok frissítése jóval könnyebben és egyszerűbben történik, elég hozzá egy speciális szoftver (amit esetleg trójai révén küldünk a rendszerbe), vagy valamilyen célhardver (ez fizikai hozzáférést igényel). De legyen szó akármilyen hardverről, valahogyan az ún. bootstrap kódnak el kell indulnia, vezérlést kell kapnia. Ezt a bootstrap kódot gyakran hívjuk firmware-nek. Ennek módosításával komoly előnyhöz juthatunk, lévén ez nem felejtő memória, újraindításkor sem törlődik ki. Azonban ez a kód nagyon hardverfüggő, és azt igényli, hogy a kódot visszafejtsük. Különösen fontos, hogy a módosítás után is megőrizze az eredeti funkciókat, mert adott esetben nem működik a készülék. Másodsorban a firmware memóriaterülete véges, túl nagy plusz kódot nem tudunk elhelyezni benne.

Egy utolsó lehetséges terület pedig a processzor mikrokódjának módosítása. Erre a modernebb processzoroknál lehetőségünk van, s így megváltoztathatjuk a CPU működését (az utasítások végrehajtását, különböző funkciók engedélyezését-tiltását). Maga ez a változtatás nem triviális feladat, s maga a frissítés akár a mikrochipet is megrongálhatja, ha nem, megfelelő módon történik. A mikrokód adatblokként a BIOS-ban tárolódik. Miért lehet hatásos egy támadásban ezt használni? Lássunk 2 példát:

- A frissítés maga letilt bizonyos funkciókat, ezáltal használhatatlanná és elindíthatatlanná téve a hardvert (DoS támadás).
- A támadó olyan módosítást injektálhatna a rendszerbe, hogy a különböző védelmi módok (Ring0-Ring3) átjárhatóvá válnának, s minden program közvetlenül kernel módban futhatna. Viszont talán a mikrokód módosítása a legnehezebben megvalósítható feladat. A mikrokód frissítés formátumát (fejlécek) ellenőrzi a BIOS, mielőtt engedélyezné. Ugyanakkor ott van a frissítés programozási nyelvének kérdése. Ezt minden chip gyártó mélységesen titkolja. A mikrokód adatfrissítések kódoltak (Intelnél), ahhoz, hogy BIOS elfogadja, ugyanazon algoritmus szerint kódolt adatblokkot kellene tudnunk előállítani. Ráadásként minden CPU máshogy kezelheti ezeket a frissítéseket.

[1], [2], [8]

3.5. Hálózati technikák

A hálózati viselkedésnek több tényezője is van. Először is ott van a rendszerbe való bejutás kérdése. Napjainkban már komoly védelmi eszközök (tűzfal, IDS,IPS, stb) védik a szervezetek magánhálózatát, melyek meglehetősen jól funkcionálnak, nem véletlenül vannak egyre kisebb méretű féregjárványok a világon. Bár természetesen belülről indítva a dolgot, máris megkerültük a legfontosabb védelmi állást, és jól tudjuk hogy a támadások 77%-a belülről indul. Ezután, ha már bejutottunk, akkor valahogy irányítani, kommunikálni kellene a bejuttatott programmal. Nagyon fontos, hogy a hálózat viselkedését monitorozó eszközök radarja alatt maradjunk. Ehhez kapcsolódik, hogy mindennek nyoma marad valahogy, általában valamilyen naplózás formájában, persze ha nem tűnik fel semmi, akkor ritkán történik kutatás, hiszen a legtöbb adminisztrátornak éppen elég rendbe tartania a rendszert, és a felhasználók különböző kéréseinek eleget tennie.

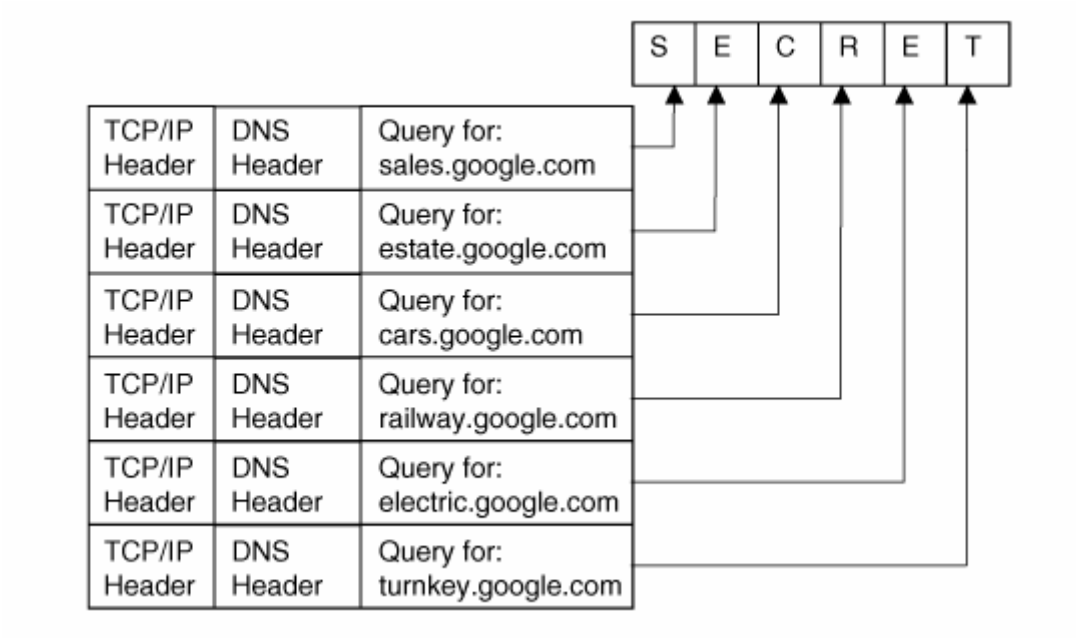
Nézzük meg egy kicsit részletesebben, hogyan kommunikálhatunk a már bejuttatott programmal. Egy direkt TCP kapcsolat szinte biztos, hogy feltűnik, minden hallgatózó program észleli. Sőt a TCP portot vissza lehet követni a folyamatig, mely használja. Ennél sokkal kifinomultabb módszerek kellenek. Egy „zajos” hálózati környezetben célszerű olyan protokollokat használni, melyből kimondottan sok található meg a hálózatban (például DNS). Ekkor olyan csomagokat kell gyártani, amelyek a legális forgalomnak megfelelőek, mégis tartalmaznak valamilyen plusz adatot. Nézzük, hogy a különböző protokollok mennyire alkalmasak erre:

- DNS (53-as port): UDP, vagy TCP csomagok egyaránt használhatóak, s általában még a vállalati hálózaton kívülre is át vannak engedve a tűzfalon keresztül.
- HTTP, vagy HTTPS (80-as, vagy 443-as port): A legtöbb esetben engedélyezett protokollok. Ráadásul az utóbbi esetben igen valószínűtlen, hogy a csomagjainkat behatóan ellenőrizni fogják (Mindenesetre azt tudnunk kell, hogy létezik technológia, mely az SSL web session dekódolására képes).
- ICMP: Általában karbantartási okok miatt a legtöbb esetben szintén engedélyezett protokoll.
- SMTP, POP3, IMAP
- NNTP
- IPSEC
- P2P
- IM
- FTP, TFTP

Célszerű a kommunikáció üzeneteit is elrejteni amellett, hogy nem tűnünk ki a hálózati forgalomból, hiszen egy kódolatlan üzenetben található nyílt jelszó bizony feltűnhet a rendszergazdának. A megoldás, hogy rejtsük el egy ártatlan kinézetű üzenetbe, vagy titkosítsuk. A szteganográfia megoldást adhat, azonban nagyon lassú lehet, ha nagy fájlokat, vagy adatbázisokat próbálunk átküldeni a hálózaton. A DNS protokollból kiindulva például lehet, hogy az egyes lekérdezések üzeneteiben rejtünk adatokat. Ha képesek vagyunk fájlokat küldeni, akkor próbálkozhatunk képekkel, vagy bármilyen más kiterjesztésű fájlokkal a „Hydan” technológiáját használva (Ez a technika képes 150-200 bájtnyi végrehajtható kód közé 1 bájtnyi titkos információt szúrni, mely aztán teljesen visszanyerhető belőle. Bár egy képbe ágyazott információ sokkal jobb tömörítést ér el (20 bájton 1 bájt adat), külön érdekesség, hogy ennek során a fájl mérete nem változik meg). A titkosítás önmaga viszont eleve gyanút kelthet (ha egy általában nyílt szöveget továbbító protokoll hirtelen hamisított karaktereket kezd tartalmazni, vagy nagy entrópiájú lesz).

Egy gyakran figyelmen kívül hagyott tényező a csomagok között eltelt idő. Egy adott protokollban a csomagokba történő üzenetkódolás helyett a maga az eltelt idő is lehet az

üzenet. Mivel a sávszélesség nagyon limitált lehet, ez csak rövid üzenetváltásokra lehet alkalmas. Egy nagyon egyszerű példa a DNS üzenetben elrejthető adatra:



7. ábra:Titkos üzenetváltás DNS protokoll használatával

A valós világban titkosítás előzi meg az üzenetet elküldését, s így kétszeres védelme is lehet, hiszen hiába fogják el, még mindig nem tudják megfejteni a tartalmát.

Megemlítenék még egy technikát, mely az előzőekre épül. Egy folyamatosan figyelő port igen feltűnő lehet. Azonban ha csak akkor működik, amikor feltétlenül szükséges ráadásul álcázza a forgalmat (például HTTP csomagoknak), igencsak megnehezíti a felderítését. Hogyan is történhet ez? Az alkalmazás figyeli a hálózati kártya interfészét, s amennyiben megfelelő tartalmú csomagszekvenciát érzékel (ezt hívják „knocking”-nak), akkor megnyitja a portot, s elvégi a szükséges kommunikációt, majd adott csomagszekvenciára le is zárja. Mint láthatjuk, ez teljesen jól beleillik az előzőleg leírtakba. Azonban ezt még egy kicsit javíthatjuk, ha nem csak a saját, hanem a hálózati szegmensre érkező csomagokkal játssza ugyanezt el. Ebben az esetben ugyanis az sem lesz egyértelmű, hogy melyik hoszt kapja a csomagokat (hiszen mindegyik szerkesztett, legálisnak látszó csomag). Persze, most mondhatnánk, hogy switcheket használó hálózatban ez nem valós veszély, de gondoljunk csak bele, hogy egyszerű „ARP poisoning” felhasználásával elérhetjük, hogy a csomagok ne csak a cél hoszthoz érjenek el.

Az anonimitást biztosító eljárásokat szintén ide sorolnám, mint a támadások egyik kritikus tényezőjét. Ez napjaink egyik legfontosabb kérdése az Interneten. Sokakat zavar, hogy

nyomon követhetők, magánéletük megsértésének tartják ezt. A szólásszabadság miatt is szükség lehet az anonimitásra, lévén egyes országok figyelemmel tartják azokat, akik egy adott téma iránt érdeklődnek, vagy különböző írásokat közölnek. Az identitás már csak azért különösen fontos, mert ha ellopják, az elektronikus világban bátran követhetnek el mindenféle bűncselekményeket, visszaéléseket, minden hozzánk fog visszavezetni. Persze létezik a másik oldal is, amely saját káros tevékenységét kívánja eltűntetni.

Proxy szerverek és láncolásuk:

Az egyik legnépszerűbb és legjobban ismert módszer. A proxy a két gép közötti pufferként működik. Ez például cégeknél egy webproxy, mely felügyeli a dolgozók hozzáféréseit, tartalomszűrést végez, gyorsítja a hozzáférést cachelés révén, stb. Virtuálisan bármely hálózati alkalmazás (Web, FTP, SSH, stb.) proxyzható. Sok vállalat kínál anonim proxy szolgáltatást, főleg internetezésre, illetve rendelkezésre áll néhány nyilvános szerver is. A fő probléma minden ilyen szolgáltatással a szerver naplók kérdése. A célgép természetesen a proxyval kommunikál, sosem látja a klienst, azonban semmi biztosíték nincs arra, hogy a naplókat törlik és nem adják át harmadik félnek (több példa is volt arra, hogy proxy naplókat adtak át bírósági kérésre, holott elméletileg azok már törölve voltak). Továbblépkedve ezen a logikán, megtehetjük hogy sok proxy szerveren keresztül folytatjuk le a kommunikációt, így egyre nehezebb lesz a nyomokat követni (főleg ha több országon keresztül történik mindez).

„Onion routing”:

Proxy szerverek és P2P hálózatok, valamint titkosítás összekapcsolását alkalmazza a technika. Ennek legnépszerűbb megvalósítása a TOR hálózat. A kezdeményező fél egy nodelistát kap a központi szervertől, a célhoszthoz vezető útvonal véletlenszerűen generálódik, és minden az útvonalban szereplő szerver csak az előző és a következő állomást ismeri. Független kulcsokkal történik a tikosítás minden pontban. A szépsége a dolognak, hogy minden pontban védett a tartalom, a forrás és a cél cím. Semelyik gép sem tudja nyomon követni a forgalmat, csak annyit, hogy kommunikáció történik. A módszer hátránya, hogy komoly teljesítmény vonatkozásai vannak a technológiának. További gyengesége, hogy az időanalízisen alapuló támadásokkal szemben védtelen: ha valaki egy alulterhelt onion-routert figyel meg, akkor láthatja a beérkező és kimenő csomagok között eltelt időt, s egymáshoz kapcsolhatja őket. Az „intersection” típusú támadás során lehetőség van arra, hogy a kieső, vagy leálló routerek alapján detektáljuk az átmenő útvonalakat. Ugyanis a megmaradó kommunikációs csatornák biztos nem ezeken a routereken keresztül kommunikáltak. „Predecessor” támadás során pedig egy onion-routert kontrolálva a támadó képes lehet megtalálni a lánc első routerét: a támadó

ugyanazon a session során figyeli meg az útvonal újragenerálását, s elég nagy mintavétel során a kezdő routert többször fogja látni, mint a többit.

[1], [2], [8]

3.6. Kriptográfia

Egy antivírus szoftver számára a legkönnyebb módszer az egyedi mintán alapuló azonosítás. Ebből kifolyólag ezek a programok folyamatosan frissülő adatbázist használnak. Ez feltételezi azt, hogy már rendelkezésre állt egy olyan program példány, ami alapján el lehetett készíteni az egyedi aláírást. Többek közt pontosan ez a baj a nem elárasztásos támadási formákkal. Mire sikerül detektálni egy példányt, addigra késő lehet. Polimorfizmus felhasználásával azonban ezt még jobban megnehezíthetjük. A vírus törzsét kódolva, melyet egy véletlenszerű (véletlen kulcsot használó) dekódoló eljárás kódol ki futási időben, új programot kapunk, mely őrzi az eredeti funkcionalitását, viszont megváltozott binárissal rendelkezik. Ez a módszer azért könnyen támadható. Például nem mindegy, hol keresi a védelmi program a sztringet. Ha memóriában, ahol általában kódolatlan és eredeti állapotban van jelen a program, akkor könnyen találhat egyezést. Másik eset, ha maga a dekódoló kód statikus, hiszen ekkor ez is alkalmas az azonosításra. Harmadrészt viselkedésfigyelést, vagy heurisztikát alkalmazva kiderülhet a program valódi viselkedése. Erre megoldást adhat a különböző regiszterek véletlenszerű használata, vagy sorrendjük felcserélése. Mivel a polimorfizmus még evvel a trükkel is elég sok támadási felületet ad a dekódoló kód korlátossága miatt, egy új módszer lépett a képbe.

A metamorf programok ahelyett, hogy kisebb módosításokat végeznének a dekódoló kódon, képesek arra, hogy az egész programot módosítsák annak replikációja során. Ennek előnye, hogy minden programpéldány gyökeresen különbözik elődjétől. Ez a technika erőteljes kódanalizálási képességet feltételez, melyet a kártékony program magába ágyazva tárol. Minden terjedés esetén valós időben végigolvassa a programot és hozza létre az új verziót. A lényeges dolog itt is az, hogy milyen változásokat lehet létrehozni a programban (beleértve a metamorf motort is)?

Utasítás és regiszter kiválasztás:
A program kódjának újragenerálása során sok paramétert megváltoztathat, például egy speciális utasítás és regiszter halmazt alkalmazhat (általában több mint egy utasítást lehet használni egy adott művelet végrehajtásához).

Utasítás sorrendezés:

Egy adott függvényen belül néha lehetőség van véletlenszerűen megváltoztatni az utasítások sorrendjét.

Feltételek újragenerálása:

A metamorf motor képes a néhány feltételes ugrás megváltoztatására, például két operandus egyenlőségének vizsgálata helyett az egyenlőtlenséget vizsgálhatja. Ezáltal erőteljes átrendeződést idéz elő a kód szerkezetében, hiszen ez a különböző feltételblokkok kényszerű átrendeződésével jár.

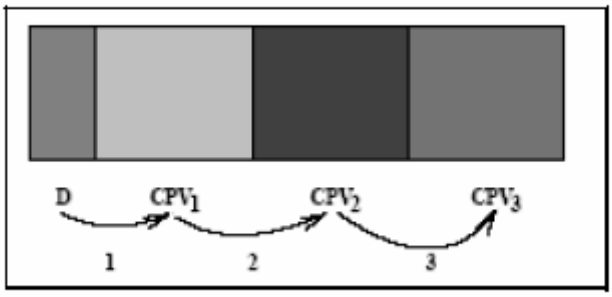
Szemét beszúrása:

Olyan utasítások véletlenszerű beszúrása, melyek a program funkcionalitásán nem változtatnak.

Függvény sorrend:

A modulban a függvények sorrendje irreleváns a program futása szempontjából, megváltozása a mintakereső motorokat mégis összezavarhatja.

Szót kell még ejteni egy olyan koncepcióról, mely kiteljesíti kriptográfia alkalmazását, s jelentősen megnehezíti a detektálást, valamint a visszafejtést is. Legyen a kikódoló kulcsa egy olyan érték, mely több környezeti adatból tevődik össze. Ezt környezeti kulcsgenerálásnak nevezzük.



8. ábra:Kódolt vírus szerkezete

Az ábrán a vírus törzsének szerkezetét láthatjuk.

- D: a kikódolásért felelős kódrész, mely ezen kívül a környezeti adatokat is gyűjti és analizálja.
- CPV1: Az első kódolt rész, melynek kulcsa K1 (miután dekódolásra került CPV1), tartalmazza a különböző anti antivírus funkciókat.

- CPV2: A második kódolt rész (CVP2), K2 kulccsal. Dekódolás után itt találhatók meg a polimorfikus függvények.
- CPV3: Tartalmazza a „payload”-ot, kulcsa K3.

A konstrukciós adat lehet például a helyi DNS cím, rendszer idő, a rendszerre jellemző adat, és végül egy olyan információ, ami teljesen a támadó birtokában van, és a rendszeren kívül található és nyílt csatornán keresztül elérhető. A dekódolási eljárás a következő:

1. A dekódolási eljárás, ***D*** összegyűjti az adatokat közvetlenül, vagy többszörös mintavétellel és kiszámítja a 160 bites értékét **V**-nek.

2. Amennyiben if $V = M$, ahol M az aktiválási kód, melyet a vírus törzse tartalmaz, akkor létrehozza a K_1 kulcsot, mely a környezeti adatok értékéből létrejött hash érték. Ellenkező esetben a vírus letörli magát a rendszerből.

3. ***D*** dekódolja az CPV_1 –et $VP_1 = DK_1(CPV_1)$ és elindítja. Ezután ***D*** kiszámolja a

$K_2 = H(K_1 + v_2)$ kulcs értékét, ahol v_2 az utolsó 512 bitje VP_1-nek.

4. ***D*** dekódolja CPV_2 -öt $VP_2 = DK_2(CPV_2)$ és elindítja. Ezután ***D*** kiszámolja a $K_3 = H(K_1 + K_2 + v_3)$ kulcs értékét, ahol v_3 az utolsó 512 bitje VP_2-nek.

5. ***D*** dekódolja CPV_3 -at $VP_3 = DK_3(CPV_3)$ és elindítja.

6. Miután a vírus utasításait végrehajtotta, teljesen eltávolítja magát a rendszerből.

Néhány megjegyzés a protokollal kapcsolatosan:

- Minden terjedés után teljesen új kódot kaptunk (beleértve a dekódoló eljárást is).
- A K1, K2, K3 kulcsok függetlenné tehetők egymástól további környezeti kulcsgenerálás révén.
- A VPi különböző részei tömörítve lehetnek a titkosítás előtt.
- A vi értékek célja, hogy az adatok az egész 512 biten eloszoljanak.
- Az automatikus törlés késleltetődhet, ha az idő és dátum értékeket kevésbé szigorúan kezeli.

Ezzel a protokollal a védett kód az analízist exponenciális problémává teszi (Feltéve, ha áll rendelkezésre mintakód, hiszen az automatikus törlés és polimorfizmus miatt ez nem könnyű feladat).

Van még egy helyszín, ahol a kriptográfia a vírusokat és férgeket segíti. Olyan kártékony kódok esetén, mely megfertőzött gép működését saját maguktól teszik függővé. Ha a vírus letörlődik, elpusztul az adat is. Ezt a megoldást teljesebbé lehet tenni olyan módon, hogy egymással kapcsolatban álló, kommunikáló víruspéldányok figyelik egymást, s detektálják valamelyikük kiesését.

Miután ilyen jól felderítettük, hogy miért nem lehet hatékony a minta alapú felismerés, nézzük át, milyen egyéb lehetőségeink vannak.

Heurisztika:

A különböző kártékony kódok nem úgy viselkednek, mint a normál programok, hiszen olyan műveleteket tartalmaznak, melyek nem átlagosak. Például egy vírus alapértelmezetten terjeszti önmagát. Egy szövegszerkesztő, vagy böngésző program nem tesz ilyet. Így minden olyan program, mely egy másik program kódját módosítja, gyanúsítható avval, hogy vírus. Persze azért vannak kivételek is. Ilyen például a patchelés, vagy a frissítések esete. Napjainkban a kódemuláción alapuló heurisztikák a legelterjedtebbek. Ekkor egy virtuális gépben történik a program futtatása, melynek során a gép CPU és memóriamenedzsment eljárásait szimulálják. Maga a heurisztika használata teljesítmény problémákat vet fel, valamint igen sok hamis pozitív találatot is ad.

Lehetőségünk van a kódban anomáliák detektálására. Például a vírusok, mint azt előzőleg is láthattuk, gyakran csatolják magukat a fájlok végéhez, írják át a belépési pontot. Ez, valamint az abnormálisan nagy ugrás a kód elejéről, szintén kimutatható.

Képesek vagyunk a viselkedési anomáliák megfigyelésére is. A fájl és hálózati műveletek figyelemmel kisérése nagy segítség lehet a kártékony kódok kiszűrésében. Egy polimorfikus program esetében azonban nem ilyen egyértelmű a dolog, hiszen azt futási időben kell megfigyelni, ami kellően erőforrás igényes lehet. Egy elterjedt megoldás erre a rendszer szolgáltatáshívásoknak a megváltoztatása, ez már viszont kernel szintű módosítást jelent. Kérdés ki lesz az első, egy rootkit-vírus vagy a védelmi szoftverünk? A heurisztika egyik hátránya lehet, hogy gyakran ad ki hamis riasztásokat, s ezzel igencsak zavarhatja a felhasználókat. Másrészt, ha a vírusban vannak kódolva a számára szükséges címek, akkor a

vírusnak egyáltalán nincs szüksége a megszakítás táblákra, s egyebekre, kétségtelen viszont, hogy ez csak nagyon rendszer specifikus programok esetében működik.

Változásmenedzsment:
A különböző rendszerváltozások pontos nyomon követése, integritásellenőrző programok, kernel pillanatképek, stb. révén.

[1], [3], [4], [7], [9], [10]

3.7. Rootkitek

A rootkitek olyan programok, melyek biztosítják, hogy a támadó továbbra is „root", azaz rendszergazdai jogosultságokkal legyen képes láthatatlanul belépni a rendszerbe. Az én értelmezésemben a rootkit egy technológia, melyet lehet jó és rossz célokra is használni. Például a védelmi (HIDS, NIDS) és a kémprogramokban egyaránt megtalálható ez a technika. De vállalatok is használhatják annak ellenőrzésére, hogy dolgozóik betartják-e az előírásokat, eljárásokat. A programok elsődlegesen távoli hozzáférésre és vezérlésre, valamint megfigyelésre szolgálnak, hiszen ahhoz hogy adatot lopjunk nem szükséges programokat telepítenünk elég, ha eltüntetjük a nyomainkat. A rootkitek által használt technikákat részletesebben kifejtettem a memória technikák résznél.

Az emberek többsége az operációs rendszerekhez kapcsolják ezt a technológiát, azonban érdemes tudnunk, hogy bármilyen szoftverbe beleépülhetnek. Vegyük példánknak az Oracle adatbázisát. Ahhoz hogy itt megvalósítsunk egy rootkitet, a következő funkcióknak kell implementálnunk: felhasználók, folyamatok elrejtése az adatbázisban. Mivel a felhasználók és szerepköreik a SYS_USERS táblában tárolódnak, elég, ha a rá vonatkozó nézetek lekérdezéseit módosítjuk, valamint szinonimákat hozunk létre az eredeti nézetek elé, melyek a már módosított lekérdezéseket tartalmazzák. A folyamatok esetén ugyanezt kell eljátszani. Két dolog miatt voltunk erre képesek. Az adatbázisban nincsen metaadatokat védő réteg, és nézeteket használnak a táblák direkt felhasználása helyett. További referencia: [17].

Elemezzük, milyen lehetőségeink vannak a rootkitek elleni védelemre. Avval a feltételezéssel élve, hogy a rootkit használja fájlrendszert, megpróbálhatjuk detektálni a jelenlétét (valamilyen fájl ellenőrző programmal, például: Tripwire). Ez persze nem hatásos abban az esetben, ha a rootkit csak memóriában van jelen, vagy már eleve rajta volt a gépen. Az utóbbi esetben ugyanis az ellenőrző program által lekérdezett adatokat már módosíthatta. A memóriában is kutathatunk a rootkit után. Először is figyelhetjük a programok betöltődését,

azaz azokat a függvényeket, mellyel egy program képes a memóriába kerülni (NtLoadDriver, stb.). Nagyon sok ilyen függvény van, ezért könnyen kihagyhatunk egyet, s máris túl jutott rajtunk a kártékony program. Másrészt a detektáló programnak a szimbolikus linkeket is meg kell figyelnie, mert egy ilyen link hívásakor az eredeti név nem jelenik meg, s így hiába hoztunk létre kampót, az nem fogja észlelni a névfeloldás hiányában. Lehetséges a memóriát időről-időre végigkeresni ismert rootkit minta után, de ez csak a már ismerteket képes detektálni. Következő lépésként a rendszerben jelenlévő kampókat kell keresnünk. Ennek általános algoritmusa, hogy a különböző táblákat (IDT, SSDT, IRP) nézzük át, s megkeressük, hogy az adott ugrással nem lép-e át egy bizonyos memóriacím határt a program (például az IAT-ban az importált függvények meghatározott memóriacímeken kezdődnek s meghatározott méretük van). Természetesem ezen határok kialakítása más és más lehet.

[8], [17]

4. Terjedési algoritmusok

Idáig elsősorban a rejtve maradás lehetőségeit elemeztem, hiszen hosszútávon ez életszerű. Ugyanis minden érzékelt támadást utólagos analízis, rekonstrukció követ, ami együtt jár az újabb védelmi intézkedések kidolgozásával, és alkalmazásával (technológiai, eljárás, stb. szinten), arról nem is beszélve, hogy a nyomok akár hozzánk is elvezethetnek. Kérdés, hogy mi a célunk a támadással. Amennyiben a kritikus infrastruktúra megbénítása (például háború esetén), vagy gyors információszerzés, akkor a gyorsan terjedő férgekkel egybekötött támadások a célszerűek. Látni fogjuk, hogy ez esetben viszont nincs védelem. Hiszen a megfelelő sérülékenységet kihasználva akár percek alatt is megtörténhet a dolog. Ez szükségszerűen jár rombolással, hiszen nem lehetünk tekintettel a rendszer épségére, sőt nem is akarjuk, hogy sértetlen maradjon. Ráadásul egy nagyméretű féregtámadás eleve olyan forgalmat generál, mely önmaga lebéníthatja a megtámadott hálózatot. Ez persze felvet egy újabb lehetőséget hiszen, ha már nagy a forgalom, könnyen eltűnhetünk a tömegben, mintha csak a saját célzott forgalmunk látszódna a tűzfal naplókban.

Milyen lehetőségei vannak egy gyorsan terjedő féregnek? A következőkben a terjedési algoritmusok lehetőségeit vizsgálom meg.

Állítsunk fel egy elméleti a határt a terjedési sebességre. Legyen ez a fény sebessége, melyre túl sok absztrakció nélkül igaz az állítás. Tételezzük fel, hogy a féregnek teljesen pontos képe van a hálózatról, valamint semmilyen időt nem fecsérel el használaton kívüli címtartományok átkutatására, vagy a nem sérülékeny gépek tesztelésére. Ebben az esetben az egyedüli, terjedést befolyásoló tényező az, hogy egyszerre hány rendszert képes megfertőzni adott t időn belül. Legyen ez a terjedési gyorsaság r. A terjedési gyorsaság a következő módon írható le, feltételezve, hogy a már eredetileg fertőzött rendszerek száma i.

$n0 = i$

$n1 = n0 + r*n0 = (r + 1)^n0$

$n2 = n1 + r*n1 = (r + 1)*n1 = (r + 1)*(r + 1)*n0 = (r + 1)^2*n0$

….

$nt = n(t-1) + rn(t-1) = (r + 1)^(tn*0) = (r + 1)^ti$

Abban a speciális esetben, ha i=1, tehát egyetlen egy hosztról indul ki a fertőzés, t = log(r+1)nt idő szükséges egy adott populáció eléréséhez. Megjegyzés: r a sikeres fertőzések

száma, ebbe nincsen beleszámolva a sikertelen, vagy újrafertőzések esete. Ebből az elméleti határból kiindulva, még a nagyon gyorsnak számító Warhol technika, és annak legsikerültebb képviselője a Slammer/Sapphire féreg a maga 10 perces idejével is eltörpülni látszik. Az r=1mp/rendszer terjedési idővel számolva láthatjuk, hogy egy egymilliós populációt 20 mp alatt fertőz meg.

Arány	Pop. 100	Pop. 1000	Pop. 10000	Pop. 100000	Pop. 1 millió	Pop. 10 millió
0.2	25.26	37.89	50.52	63.15	75.78	88.40
0.5	11.36	17.04	22.72	28.39	34.07	39.76
1	6.64	9.97	13.29	16.61	19.93	23.25
2	4.19	6.29	8.38	10.48	12.58	14.67
5	2.57	3.86	5.14	6.43	7.71	9.00
10	1.92	2.88	3.84	4.80	5.76	6.72
20	1.51	2.27	3.03	3.78	4.54	5.29

9. ábra:Elméleti terjedési sebességek adott populáció esetén

Elméleti fertőzési sebességek különböző fertőzési gyorsaság esetén. Bár a feltételezett tudás az adott hálózatról igen csak maximalista, a Warhol vagy a Flash terjedése szépen közelíti ezt az eredményt, a teljes hálózat előzetes szkennelését használva.

4.1. Warhol

A féreg terjedése során az egyik előzetes probléma, hogy viszonylag lassan terjed, míg eléri a mérföldkőnek számító 10000-es populációt. Ennek legyőzése viszonylag egyszerű, erre szolgál a az előzetes találati lista létrehozása, melyet még a féreg útnak indítása előtt hozunk létre. Ez a potenciálisan sérülékeny, lehetőleg jó hálózati kapcsolatokkal rendelkező gépekből áll. A kezdeti fertőzés után a féregpéldányok felosztják maguk között a listát, így gyorsítva a keresési-fertőzési fázisokat. A kezdeti lista létrehozására különböző módszereket használhatunk, például elosztott, rejtett szkennelés, DNS elkérdezések, webspiderek. További hátráltató tényező lehet az algoritmus, amivel a féreg meghatározza a további célpontokat. A véletlenszerű hálózati szkennelés esetén egy-egy címet gyakran többször is megvizsgál, valamint nem tudja hatékonyan meghatározni, hogy az összes sérülékeny gép meg van-e fertőzve. Erre ad megoldást a partícionált permutációs szkennelés, mely során minden féreg példány az IP tartomány egy pszeudo-random permutációját osztja fel egymás között. Ez könnyen generálható egy 32 bites blokk kódoló és egy előreválasztott kulcs segítségével: egyszerűen kódoljuk az indexet, hogy megkapjuk a címet, és dekódoljuk a címet, hogy megkapjuk az indexet. Ha egy gép megfertőződik, szkennelni kezd a permutációjában lévő

ponttól, illetve a felosztja a felére a permutációs teret, s az új gép lesz felelős a tér feléért. Ha egy már megfertőzött gépet talál, úgy egy új random, vagy szekvenciában rögzített pontról kezdi újra a szkennelést, hiszen tudja, a már fertőzött példány felelős a tér következő részének átkutatásáért, és már előrébb is tart ebben. Ha féreg egy példánya sokszor talál fertőzött gépet anélkül, hogy egy fertőzhetőre is rátalálna, feltételezi, hogy már minden sérülékeny gép megfertőződött és abbahagyja a keresési fázist. Egy beépített időzítővel ellátva azonban időről- időre életre lehet kelteni a féregpéldányokat, s ellenőrizni, nincsenek-e újonnan a hálózatra kapcsolt sérülékeny, vagy újratelepített, de még nem frissített gépek. Az algoritmus nagyon jól működik együtt több sérülékenység vizsgálatával is: amennyiben úgy érzi, hogy az adott sérülékenységgel már nem találhat további fertőzhető gépeket, újabb sérülékenységre áll át, beállítja az aktuális címére a permutáció értékét, megváltoztatja a permutációs kulcs értékét, s újrakezdi a működését. Úgy tűnhet, hogy az algoritmus megtéveszthető, hiszen, ha egy fertőzetlen gép képes úgy válaszolni a féregnek, mintha már meg lenne fertőzve, akkor képes az utána következő címteret megvédeni a szkenneléstől. Mivel azonban a permutáció minden újraszkennelésnél változik, ezért azon gépek halmaza, melyek védettek maradnak, folyamatosan változik. Tehát ez a védelem csak akkor működőképes, ha nagyszámú gép képes a fertőzöttség látszatát kelteni.

4.2. Flash

Ez a módszer nagyon hasonló az előző technikához, annyiban azonban mégis különbözik, hogy sokkal szélesebb skálájú előszkennelését végez, lényegében az egész sebezhető webszerver címteret feltérképezve, s mintegy 9 millió hosztot tartalmazó címlistát építve. Ez a lista az Internet nagy részét lefedné, s tömörített formában mintegy 7,5 Mbyte nagyságú lenne. A kezdeti fertőzési szakaszban ezt a címlistát n részre felosztva minden féregpéldány a saját blokkjában található első címet fertőzi meg (vagy pedig egy kellően nagy sávszélességgel rendelkező hosztot). Majd pedig a gyerekféregnek átadja ezt a listát, melyet az a már előbb leírt módon újra feloszt. Ahogy a féreg terjed egyre kisebb a találati lista, amit kezelnie kell. Kutatók szerint ez az algoritmus képes lehet fél perc alatt végigfertőzni az Internetet. Azonban ne felejtsük el, hogy a találati lista létrehozása és kezelése sávszélességi gondokat okozhat.

A topológikus szkennelés, mely a fertőzött gép adatain alapszik (email címek, url-ek, P2P lista) a kezdeti szakaszban segítségére lehet a féregnek, mielőtt átváltani a találati listás vagy

permutációs terjedésre, révén sokkal kisebb sávszélességet foglal, és valószínűsíthetően kevesebb eredménytelen próbálkozása lesz.

Az összes eddigi terjedési algoritmussal (permutáció, random, teljes-találati lista) idáig az volt a baj, hogy a féreg példányai nem koordinálták egymás közt a terjedésüket, s így bizony előfordult a terjedés során redundancia. Az igazi nehézség egy alacsony költségű kommunikáció kialakítása. A P2P hálózatok hasonló nehézséggel küszködnek, így az ottani algoritmusok használata hatékony lehet. Az ACHORD algoritmust használva a férgek O(logn) idő alatt képesek egy másikkal kommunikálni, s ugyanennyi más féreg példányról kellene hatékony információt tárolni a legoptimálisabb kapcsolattartás érdekében. [20]

Az algoritmusok vizsgálatával láthatjuk, hogy a gyors terjedéshez, mire van szüksége a féregnek: gyors, állapotmentes szkennelési algoritmus, Egy már ismert célpont lista, az újrafertőzések elkerülése, párhuzamos feldolgozási lehetőség, valamint kicsi kódrészlet, és gyors fertőzés. Idáig nem volt ugyan róla szó, de még azt is ki kell emelni, hogy ha többszálú program végzi a szkennelés, fertőzés végrehajtását, akkor lényegesen gyorsabbá válhat az egésznek a működése.

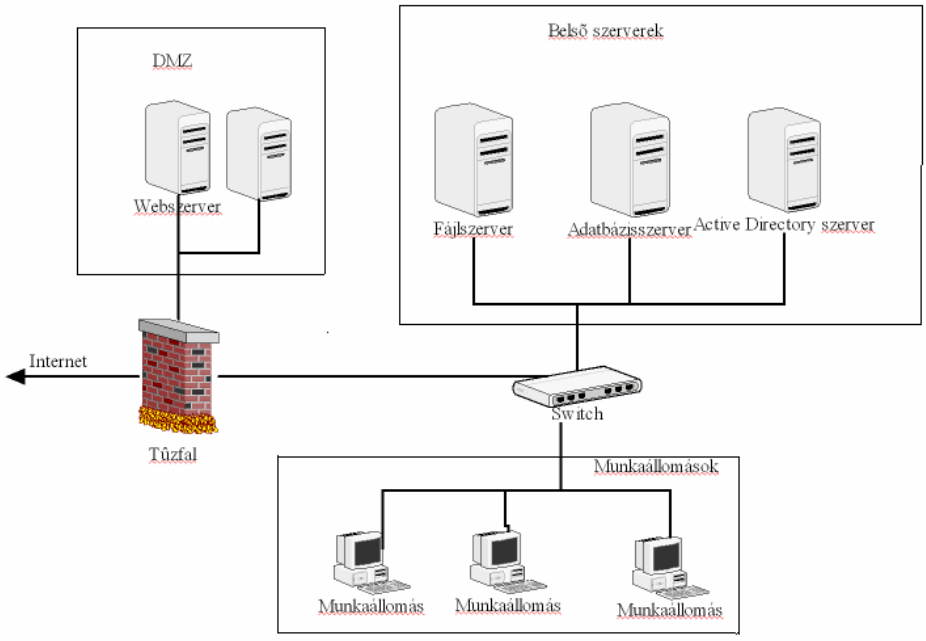
Egészen idáig nem volt szó másról, csak kizárólag a féreg terjedésének vonatkozásairól. Most nézzünk egy kicsit bele abba, hogy milyen akciókat hajthat végre a fertőzés után. Lehetőség van pusztításra, a megfertőzött gép használhatatlanná tételére. Ezzel azonban magát a terjedést lassítaná le. Kétségtelen, hogy ha evvel az akcióval addig vár, amíg a fertőzöttség el nem éri a közel 100%-os szintet, már nem akadályozza önmagát a féreg. Ezt viszont önmagának kell felbecsülnie, révén, hogy együttműködés nélkül nem kaphat globális képet.

[11], [12], [18], [20]

5. Támadási koncepciók

Ebben a részben elemezni kívánok az eddig felsorolt technikák, és algoritmusok segítségével két támadási módot. Az egyik természetesen egy nagyobb és gyorsabb féregtámadás lesz, mely amellett, hogy adatokat szerez, megbénítja a szervezet infrastruktúráját is. A másik pedig egy lassabb, de sokkal kevesebb nyomot hagyó támadás, mely célja információszerzés lehet (hiszen ha egyszer hozzáférést szereztünk a rendszerhez, akár rombolhatunk is).

A célpont mindkét esetben ugyanaz a szervezet, mely közepes nagyságú számítási infrastruktúrával rendelkezik, főként a Windows operációs rendszer különböző változatait használva. A szerverek külön szegmensben vannak, természetszerűleg tűzfalak között helyezkednek el, demilitarizált zónát alkotva. A munkaállomások főleg Windows 2000-ek és XP-k. Lássuk az első esetet.



10. ábra: A tesztrendszer ábrája

Ekkor gyors támadás esetén a reagálás lehetetlen a fertőzési időn belül (nem lehet lezárni a rendszert, vagy átirányítással élni). Szükség van először is megfelelő sérülékenység listára, valamint a hálózat feltérképezésére. Erre több módszer is adódik, itt a teljesség igénye nélkül említenék párat: különböző szkennelési formák, melyek elosztott voltukból adódóan az IDS-ek, IPS-ek detektálási szintjei alatt maradnak (Tcp Stealth Scan - nmap), google hack: keresés

a „nagytestvér” segítségével, a kívülről elérhető szolgáltatások és azok hibáinak feltérképezése, felderítő féreg használata (metamorf motorral, annak elkerüléseként, hogy ugyanazon minta sokszori ismétlődése feltűnjön a védelmi szoftver számára), stb. A férgek bejutási módszerének kiválasztása jön. A lehetőségek széleskörűek, lehet emailt használni (amit napjaink email férgei egyre csökkenő sikerességgel tesznek meg), IM támadás (feltéve, hogy engedélyezve van ez a szoftverfajta a biztonságpolitikában), böngészők hibáinak kihasználása, a különböző szervereken felfedezett lyukakat kihasználni, egyszerűen egy laptopot csatlakoztatni a belső hálózathoz, vagy WiFi-n keresztül becsatlakozni s azon keresztül indítani. Talán a legáltalánosabb eset, amikor valamely nyilvános szerver hibáját kihasználva átvettük felette az irányítást (ez nem feltétlenül ugyanaz a sérülékenység, melyet a féreg használni fog). Ekkor következik az a fokozat, amikor útjára indul a féreg. Ezt a valószínűsíthetően nagy hálózati forgalom miatt viszonylag hamar fogják észlelni, ezért nagyon fontos a gyorsaság. A különböző terjedési algoritmusok közüli választást nagyban befolyásolja az, hogy milyen ismeretek állnak rendelkezésünkre a hálózatról. A topológikus, valamint a Warhol terjedési technika abban az esetben is sikeres és gyors terjedést biztosít, ha a szervereken kívül nincsenek további ismereteink a hálózatról (hiszen ekkor nem tudunk találati listákra hagyatkozni). Férgünk többféle taktikát is alkalmazhat annak érdekében, hogy a kiválasztott szervezet entitásait támadja meg, s a kívülállókat mindenképpen megkímélje, még akkor is, ha valahogy kijutott a vállalati hálózatrészből (ez a tulajdonság idáig alapvetően nem volt jellemzője az Internetet általánosságban támadó férgeknek). Kihasználhatja az IP cím kiosztást, hogy a szkennelést a szervezet címtartományára korlátozza. A top level domain beállításával, valamint különböző nyelvi beállításokkal tovább csökkenthető a féreg működése. Végezetül a féreg szabadon terjedhetne, de működésbe csak a kiválasztott IP címtartományon belül lépne (ennek megvan az a hátránya, hogy az észlelés valószínűsége megnő, s hamarabb jut el vírusvédelmi cégekhez a befogott kód). A vállalati infrastruktúrán belül a teljes fertőzés nagyon gyorsan végbemehetne. A 100 Mbps, és az 1Gbps hálózatokban egy új fertőzés kevesebb, mint egy másodpercet igényelne. Mivel a féreg terjedése exponenciális görbéjű, a teljes fertőzöttséghez kevesebb, mint egy perc kellene. Ráadásul a nagy számú sérülékeny hoszt maga is gyorsítja a terjedést.

Nem beszélhetünk 100%-os lefedettségről a fertőzés tekintetében, lévén néhány alhálózat, vagy hoszt el lehet zárva a teljes belső hálózattól, vagy csak lekapcsolt állapotban van, esetleg ellenálló a féreg által hordozott sérülékenységgel szemben, vagy az operációs rendszer olyan verziójával rendelkezik, mely már rendelkezik a hiba foltozásával. Ezt ellentételezheti, ha

valamilyen „zero hour” sérülékenységet használ a féreg, illetve, ha sok sérülékenység van beprogramozva a féregben. Ez utóbbi viszont azzal jár, hogy a terjedési idő megnövekszik, vagy többszörös fertőzés következik be (menedzselési komplexitása nő). Tételezzük fel, hogy a legrosszabb esetben az egész számítógépes állomány 60%-os fertőzöttsége történhet meg. 2000 gépes hálózatnál ez 1200 gépet jelent. Miután a féreg úgy értékeli, nincsen több fertőzésre alkalmas gép, elkezdheti működésének második fázisát, az információgyűjtést, majd a gépek rongálását, például merevlemezek törlése, felülírása, adatok tikosítása révén, akár hardveres támadás is történhet, például a BIOS újraírásával (bár ez gyártófüggő, azonban automatizált analízissel kideríthető a BIOS frissítés megvalósíthatósága, amely része lehet a féregnek). Nézzük a helyreállítás költségét. Tegyük fel, hogy az adminisztrátor számára 1 órát vesz igénybe a rendszer helyrehozása (tömeges telepítés, és a számítógépek párhuzamosan történ kezelése révén). Másodlagos veszteségnek tekinthető a kiesett munkaidő által okozott anyagi kár. Végül harmadsorban az elvesztett adatok, információk okozta kárt kell felmérni. A következők viszont mindenképpen megfontolásra adnak okot: Milyen biztosíték van arra, hogy található jól kihasználható biztonsági lyuk a dedikált rendszerben (tömeges fertőzésre ad lehetőséget)? Feltételezve, hogy naprakész, és jól menedzselt rendszerről van szó ennek viszonylag kicsi is lehet a valószínűsége. Előbb említésre került, hogy „zero hour” sérülékenység esetén a féreg sikeres működésű lehet. Ekkor azonban nem számoltunk avval, hogy egy ilyen sérülékenység felfedezése nem biztos hogy akkor következik be, amikor a támadónak szüksége lenne rá. Ráadásul a kellően inhomogén rendszerkörnyezet is gátló tényező lehet. Természetesen ott van még annak az átfutási ideje is, amíg az új biztonsági frissítéseket telepítik (először: tesztrendszerben történő telepítés, később a teljes rendszer frissítése).

Véleményem szerint az Internet teljes egészét tekintve sokkal nagyobb hatású lehet egy gyors féreg. Terjedését és hatását tekintve, a nagy számok törvénye alapján itt sokkal inkább található védelem nélküli gép, mint egy kisebb hálózaton. Egyetlen hálózatot elemezve azonban igen sok gátló tényezőt találhatunk.

Azt az esetet véve alapul, amikor megfigyelésre, s adatszerzésre szorítkozunk, könnyebb dolgunk lehet. Ekkor szintén szükség van a hálózat, valamint sérülékenységeinek felderítésére, amely az előzőleg ecsetelt módokon keresztül történhet. Innentől azonban megváltozik a módszer. Mivel a cél az észrevétlen behatolás, nem áraszthatóel a hálózat mindenféle programmal, hiszen ha más nem, akkor ez mindenképpen detektálható. A cél, hogy a hálózat olyan kritikus pontjain legyenek elhelyezve a különböző „detektor”

alkalmazások, melyek az átmenő információkat szűrni, s továbbítani tudják. Elsősorban a szerverekre kell koncentrálni, révén, hogy az adatok központi tárolása itt megoldott. Az átmenő adatok figyelése promiscuous módú hosztok esetén túlzottan feltűnő lenne, még a különböző rootkit technológiák alkalmazása esetén is (a rootkit technológiák más gépről való tesztelés és szkennelés esetén nem védenek, mint ahogy tiszta forrásból betöltött rendszer esetén is jól láthatóak az immár nem rejtett futó folyamatok, portok).

Jelenleg a legvalószínűbb támadási koncepciót a következőképpen tudom elképzelni, avval a feltételezéssel élve, hogy a célgépekhez nincsen fizika hozzáférés. Ebben az esetben ugyanis célszerűbb a hardveres rootkitek használata, révén sokkal kevesebben gondolnak rá, hogy ilyet keressenek, másrészt a gépek fertőtlenítése sem triviális feladat, legvégül pedig nem kell program betöltődésével foglalkozni. Social engineering technikák alkalmazásával lehetőséget találhat a támadó ilyen telepítésére, vagy pedig a már régóta jól bevált taktikát alkalmazva megfertőzheti a hardverfrissítésre szolgáló binárisokat (firmware frissítések, stb.).

De nézzünk egy általánosabb esetet, amikor nincsenek ilyen lehetőségek. A programok célpontjaként itt is a szerverek a legalkalmasabbak, lévén rajtuk tárolják a kritikus adatokat. Nem mindig szükséges az operációs rendszer fölött átvenni az irányítást, néha elég a szerveralkalmazást módosítani (adatbázis „rootkit”). A támadó saját maga rejtésére használhatja a már említett anonimitást biztosító technikákat (proxy láncolás, onion-routing, stb.).

Ezután valamilyen módon be kell jutnia a rendszerbe. Ehhez használhat valamilyen sérülékenységet, vagy pedig akár a belső hálózatról is kapcsolódhat. Miután kapcsolódott, fel kell telepítenie a megfigyelő és adatgyűjtő programokat. Ezek a programok olyan trójaiak, melyek rendelkeznek rejtőzködő tulajdonságokkal. Viszont fontos, hogy minél kevésbé tűnjenek fel viselkedésükkel, ezért a lehető legkevesebb rejtési funkciót kell végezniük (ugyanis minél több adatot rejt el, másít meg, annál valószínűbb, hogy egy adott lekérdezés páros különbséget fog találni). Kétségtelen, hogy futásuk során el kell rejteniük a folyamatukat az operációs rendszerben. Ehhez egy DKOM rootkitre van szükség, mert a felhasználói módúak könnyen észlelhetőek, másrészt a rendszerben jelenlévő kampók könnyen kideríthetőek lekérdezéssel. Optimális esetben, a rootkit képes beépülni valamilyen fájlba, mert így nem kell magát fájlt elrejteni az operációs rendszer elől (amit lényegében nehezebb is, mint egy folyamatot), valamint ha ez egy állandóan lefutó alkalmazás fájlja, akkor a rootkit betöltése is garantált a rendszer indulásakor. Ezenkívül használhatna registry kulcsokat vagy ini fájlokat (Windows esetén), drivereket a betöltéshez, vagy adott esetben

módosíthatja a bootloadert is. Az is fontos kérdés, hogy hol tárolja a megszerzett adatokat, illetve hogyan kommunikál kifele. Az adatok tárolásához valamilyen fájlra van szükség, hiszen a memória véges, s amúgy is feltűnne ekkora erőforrás kiesés. Célszerű egy létező fájlban kódoltan tárolni az adatokat, s így is továbbítani adott esetben (kisebb méret, valamint elfogás esetén sem tudódik ki a tartalma). Kisebb méretű adatokat akár Hydan technika alkalmazásával is rejthetünk és tárolhatunk. Legvégül elérkeztünk az egyik legkritikusabb kérdéshez. Hogyan kommunikálhatunk? Az állandó figyelő portnál nincsen feltűnőbb jelenség a hálózatban. Egy egyszerű szkenneléssel lelepleződik a tény, hogy az adott hoszt kompromittálódott. A szintén említett „knocking" eljárás csökkenti a felfedezés veszélyét. Ehhez csak olyan protokollt találni, mely legális az adott hálózati szegmensben és olyan tartalmat mely nem fordulhat elő, nehogy véletlenül aktivizálódjon a hátsó kapu. A hálózati technikáknál említett eljárásokat itt kell alkalmazni. Legvégül figyelni kell arra, hogy a kifelé irányuló kommunikáció mennyisége se legyen kirívó.

Most pedig következzen konkrétan a két eset szimulációjának leírása a fentiek alapján. A hálózat Windows 2000 Service Pack 4 operációs rendszerrel telepített gépekkel rendelkezik, mind a munkaállomások, mind a szerverek esetén, melyek még az augusztusi frissítéseket nem tartalmazzák, azaz futtathatóak rajtuk az MS05-039 sérülékenységet kihasználó kódok. A gyors féregtámadást tehát egy „zero hour" sérülékenységgel tudom szimulálni. Ezenkívül kisebb számú más operációs rendszerű hoszt is jelen van, így az is megvizsgálható, hogy több sérülékenység (LSASS, stb.) egymás utáni szkennelése milyen többletmunkával jár.

Érdekesebb lehet a másik eset megfigyelése. A következő programokat választottam ki:

- Rootkit: FU rootkit, mely megfelel a kívánalmaknak, device driverként tölthető be, és egy alkalmazással vezérelhető a működése. Így az is megvalósítható, hogy ne folyamatosan fusson, hanem szinkronban a hátsó kapu programként választott Sadoor alkalmazással.
- Hátsókapu: Sadoor, képes a hálózati interfész figyelésére, s csak meghatározott csomagkombinációra nyitja meg a portot.
- Adatgyűjtő szoftverek: regdump, Nullsoft szoftverek, Dsniff.
- Hálózatfigyelő program: Ethereal.

A mérés során a különböző funkciókat 1-1 programmal fogom helyettesíteni, természetesen a valós életben mindezt 1 program is megvalósíthatná. Az eredmények fényt derítenek arra,

mennyire detektálható egy szakaszos működésű rootkit, melyet többféle védelmi szoftverrel is tesztelni fogok. A hálózati forgalom mérése is fontos része a szimulációnak, megfejthető belőle, hogy különböző automatikus loganalizáló programok képesek-e kinyerni a kémprogramok adatforgalmát.

A két szimuláció jelen pillanatban is folyamatosan történik, eredményeiről a konferencián tartott előadásban számolok be.

[16], [21]

6. Továbblépési lehetőségek

Legvégül pedig tekintsük át, hogy az itt leírtak hova fejleszthetők tovább, milyen megfontolásokra adnak lehetőséget. A dolgozat folyamán sok rendszerismeretet adtam át. A mérés során pedig egy több-komponensű kémszoftver szimulációjával dolgozom. A közeljövőben lehetőség lehet ennek a szoftvernek egy komponenssé való integrálására, valamint olyan eljárás kidolgozására, mely képes ezt a koncepció programot megbízható módon detektálni.

7. Rövidítések, fogalmak

IM: Instant Messaging – azonnali üzenetküldő szolgáltatás (például ICQ, AOL, MSN, YAHOO)

P2P: peer to peer – lényege, hogy az informatikai hálózat végpontjai közvetlenül egymással kommunikálnak, központi kitüntetett csomópont nélkül. Ez hibatűrőbb felépítést, skálázhatóságot jelent.

Zero hour: Olyan frissen felfedezett sérülékenység az adott szoftverben, melynek jelenleg nem létezik a hibajavítása

Botnet: Olyan megfertőzött gépek, melyek tartalmaznak egy szerver programot, mellyel beléphet rájuk a támadó, és irányíthatja őket

8. Irodalom

[1] Peter Szor: The Art of Computer Virus Research and Defense, 2005.

[2] Ed Skoudis, Lenny Zeltser: Malware Fighting Malicious Code, 2004.

[3] Szappanos Gábor: A számítástechnika sötét oldala, 2003.

[4] Adam Young, Moti Young: Malicious Cryptography: Exposing Cryptovirology, 2004

[5] Virus Encyclopedia, http://www.viruslist.com

[6] http://www.f-secure.com

[7] http://vx.netlux.org

[8] Greg Hoglund, James Butler: Rootkits, subverting the Windows kernel

[9] Eldad Eilam: Reversing: Secret of Reverse Engineering, 2005

[10] Major Eric Filiol: Strong Crypography Armoured Computer Forbidding Code Analysis: the Bradley virus, 2005

[11] Stuart Staniford, Vern Paxson, Nicholas Weaver: How to Own the Internet in Your Spare Time, 2002

[12] Bruce Ediger: Simulating Network Worms, 2004

[13] http://www.securityfocus.com

[14] http://www.megasecurity.com

[15] http://www.packetstormsecurity.com

[16] Nicolas Weaver, Vern Paxson: A Worst Case Worm, 2004

[17] Alexander Korbust: Database rootkit, 2005

[18] Ivan Balepin: Superworms and Cryptovirology: a Deadly Combination

[19] Simone McCloskey: Cryptography and Viruses, 2005

[20] Brandon Wiley: A Curious Yellow, The First Coordinated Worm Design

[21] Mariusz Burdach: Practical assigment, 2004

[22] Hydan http://www.crazyboy.com/hydan

[23] Veszprog Kft: Vírusvédelmi technikák vizsgálata, 2002