Rootkits

Miha Pihler, maj 2007

Uvod

V zadnjih nekaj letih se je število velikih izbruhov virusov precej zmanjšalo. Razlogov za to je več. V tem času se je močno povečala varnostna osveščenost že pri samem razvoju operacijskih sistemov, ne smemo pa pozabiti tudi na končne uporabnike, ki so se navadili rednega nameščanja popravkov ter uporabe varnostih mehanizmov kot so požarni zidovi ter protivirusna zaščita.

Omeniti pa moramo tudi druge razloge. Vedno več napadov na omrežja je naročenih in pri napadih, kjer je cilj kraja podatkov, so svojo novo vlogo našli avtorji virusov, ki so se prelevili v avtorje rootkitov, ob tem pa, namesto da povzročajo kaos na internetu, celo kaj zaslužijo.

Primarna naloga rootkitov ni uničevalne narave ali neposredno povzročanje kaosa. Naloga rootkitov je nepridipravom omogočiti kar se da dolgo neopazen dostop in nadzor nad sistemi, kamor jim je uspelo rootkit namestiti.

Načini okužbe

Najlažji način okužbe so napake v operacijskih sistemih, pa tudi v aplikacijah. Za okužbe so zanimivi tudi vsi servisi, še posebej kadar tečejo s pravicami sistema ali (lokalnega/domenskega) administratorja.

Pogosto se sprašujemo o varnostni operacijskega sistema in pri tem prepogosto pozabljamo na varnost vse dodatne programske opreme, ki jo na računalnik namestimo – vključno z npr. gonilniki za tiskalnik, ki se namestijo kot del jedra (kernela).

V primeru, da operacijski sistem v določenem okolju ni ranljiv, pa morda lahko uporabimo kakšno ranljivost v aplikacijah, ki so v nekem okolju nameščene. Znano je, da je kar nekaj proizvajalcev programske in strojne opreme (Apple[1]) na tržišče poslalo produkte, ki so bili okuženi z virusi.

Kar nekaj uporabnikov se je z »rootkiti[2]« okužilo preko legitmno kupljenih izdelkov. Spomnimo se rootkita, ki ga je kot del DRM (Digital Rights Management) uporabljal

[1] http://www.apple.com/support/windowsvirus/
[2] Sonyev program je uporabljal tehnike skrivanja, ki so značilne za rootkite.

Sony[3]. Ta sicer ni imel destruktivne vloge, je pa učinkovito skrival vse datoteke, mape, sistemske registre in procese, ki se jim je ime pričelo na »sys«. Nevarnost je v tem primeru izhajala predvsem iz tega, da bi se lahko pojavil virus ali druga zlonamerna koda, ki bi se imenovala npr. »sysvirus.exe« z istoimenskim procesom ter ključem v sistemskem registru. Kljub morebitni nameščeni protivirusni programski opremi bi tak virus ostal skrit.

Zelo podobno tehniko skrivanja je dolgo časa v svojo programsko opremo vgrajeval tudi Symantec[4].

Zaščita pred zgoraj omenjenimi ranljivostmi je lahko relativno preprosta. Veliko več problemov pa se bo pojavilo, ko bodo bolj razširjeni rootkiti, ki se lahko skrivajo v strojni opremi npr. v BIOSu[5] računalnika ali kakšne druge opreme, priključene na računalnik. BIOS ima v današnjih računalniških sistemih precej prostora in se vanj brez težav skrije rootkit dolžine 1500 bytov. Če to ni dovolj, pa lahko prepiše katere od manj pomembnih funkcij v BIOSu (novi operacijski sistemi uporabljajo zelo omejen nabor funkcij BIOSa in verjetno ne bi nihče opazbil, če bi pobrisali vse tiste, ki niso v uporabi). Obstoječa protivirusna zaščita v takih primerih odpove, saj ni narejena za odkrivanje tovrstnih okužb z zlonamerno kodo.

Seveda ne smemo zanemariti tudi socialnega inženiringa, kjer so napadalci omejeni izključno s svojo domišljijo in iznajdljivostjo. Ti poskusijo pretentati končne uporabnike, ki so pogosto najšibkejši člen v varnostni verigi, da poženejo »trojanskega konja«, ki okuži njihov sistem.

Še zmeraj ostaja zelo pomembna fizična varnost računalniških sistemov. Večina naprav, vključno z usmerjevalniki, osebnimi računalniki, strežniki itn. je zelo dovzetnih za napade, kjer imajo napadalci (kriminalci) fizični dostop.

Za Microsoftove operacijske sisteme poznamo kar nekaj t.i. »live CD« programov, ki med drugim omogočajo resetiranje gesla skrbniškega računa, brez da bi predhodno vedeli karkoli o sistemu. Eno od bolj popularnih tovrstnih orodij je ERD Commander[6], ki poleg resetiranja gesel omogoča tudi dostop do sistemskega registra, datočnega sistema itn. Manj znano orodje je eEye BootRoot[7].

Če imajo kriminalci dostop do strežnikov, kamor so namestili rootkit, je verjetno, da imajo dostop tudi do druge opreme npr. do že omenjenih usmerjevalnikov, IDS, IPS sistemov in morda celo do požarnih zidov. Dostop do te opreme lahko napadalcem zelo koristi, saj jim lahko omogoči, da popravijo pravila ali pa definicije, kar bi lahko rootkitu omogočilo nemoteno delovanje na omrežju. Tudi če napadalec ne pozna gesla za dostop do teh

[3] http://blogs.technet.com/markrussinovich/archive/2005/10/31/sony-rootkits-and-digital-rights-management-gone-too-far.aspx
[4] http://www.eweek.com/article2/0,1895,1910077,00.asp
[5] http://www.securityfocus.com/columnists/442
[6] http://www.winternals.com/Products/AdministratorsPak/Default.aspx
[7] http://research.eeye.com/html/Tools/download/eeyebootroot.zip

sistemov, ima večina teh naprav t.i. »break sequence«, ki omogoča polni dostop do naprave tudi brez gesla. Navodila za najbolj pogosto uporabljeno opremo je možno najti na internetu.[8]

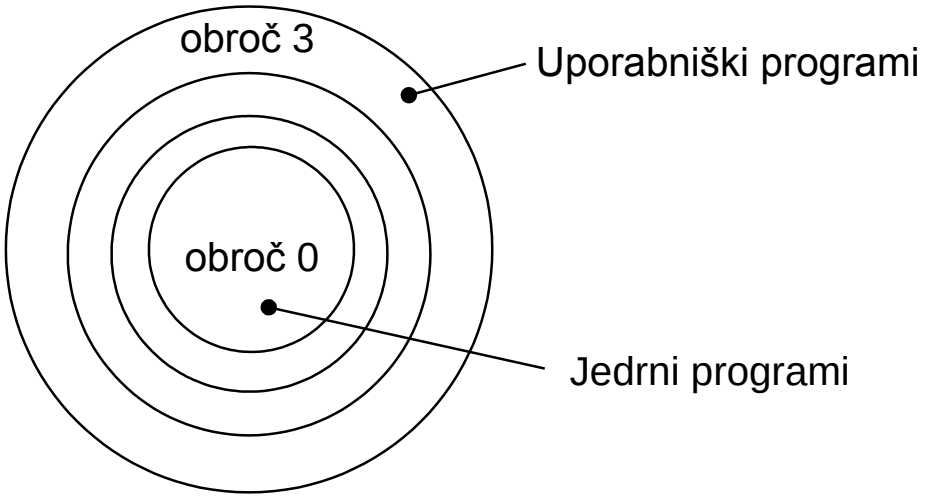
Tudi drugi operacijski sistemi niso nič manj ranljivi na fizične napade. Tako Mac OS X[9] kot Linux[10] poznata t.i. »Single user mode«, kjer je možno (v večini primerov) s fizičnim dostopom dobiti polni dostop do operacijskega sistema brez predhodnega poznavanja skrbniškega gesla.

Delovanje in skrivanje rootkitov

Uporabniški in jedrni način[11]

Intel x86 procesorji uporabljajo t.i. obroče (rings) za nadzor dostopa med programi, ki tečejo v uporabniškem načinu ter programi, ki tečejo kot del jedra. Ti obroči so implementirani na procesorju – vendar ne gre za fizično implementacijo, temveč za navidezno (»virtualno«). Kljub temu, da omenjena arhitektura implementira 4 obroče, večina operacijskih sistemov (tudi Linux in Windows) uporabljata samo obroč 0 za vso jedrno kodo (vključno z gonilniki) ter obroč tri za vse uporabniške programe (npr. calc.exe, winword.exe, ...). Programom je dodeljen določen obroč in ti privzeto ne morejo direktno dostopati do programov v nižjem obroču. Obstaja pa nekaj načinov, kako iz obroča tri vpeljati kodo v obroč nič. Tipični primer je namestitev gonilnikov.

Obroča 1 in 2 v večini primerov nista v uporabi. Vloga procesorja je, da sledi kodi in dodeljenemu pomnilniku, ki je dodeljen vsakemu obroču.



[8] Password Recovery Procedures http://www.cisco.com/warp/public/474/index.shtml
[9] http://docs.info.apple.com/article.html?artnum=106388
[10] http://www.redhat.com/docs/manuals/linux/RHL-7.3-Manual/custom-guide/s1-rescuemode-booting-single.html
[11] Angleški prevod za lažje razumevanje: »User and Kernel mode«. V literaturi pogosto tudi »user land« oz. »kernel land«

Poleg omenjenih omejitev pri dostopih med poseameznimi obroči obstaja tu še nekaj navodil (instructions), ki so privilegirane in se lahko izvajajo samo v obroču 0. Tipično so ta navodila uporabljena za direktno komunikacijo s procesorjem in drugo strojno opremo. Primer takih navodil je:

- cli – »stop interrupt processing (on the current CPU)«
- sti – »start interrupt processing (in the current CPU)«
- in – »read data from a harwarde port«
- out – »write data to a hardware port«

»Kernel Hooks«

Jedrni pomnilnik se običajno nahaja na območju 0x80000000 ali višje. Če je sistem zagnan z boot.ini opcijo /3GB, kar omogoča procesom dostop do 3GB pomnilnika, potem se jedrni pomnilnik nahaja na 0xC0000000 in višje.

Procesi ne morejo direktno dostopati do jedrnega pomnilnika razen kadar imajo pravico `SeDebug` (debug) in uporabljajo API klice za razhroščevanje (debugging). To pravico imajo privzeto vsi procesi, ki jih požene sistemski skrbnik (administrator) oz. uporabnik, ki je član skupine »local / domain administrators«.

Jedrni rootkiti navadno delujejo kot gonilniki za naprave (device drivers) in jih je kot take najlažje implementirati.

Za avtorje rootkitov obstaja veliko dobrih razlogov, zakaj rootkit skriti v jedro. Poleg očitne vrednosti – poln dostop do sistema jim obroč 0 nudi tudi dobro zaščito. Kar nekaj programov za odkrivanje zlonamerne kode teče v obroču 3 in se jim rootkit, ki teče v obroču 0, brez težav »skrije«. Tudi če program za odkrivanje zlonamerne kode teče v obroču 0, ima rootkit, ki ravno tako teče v obroču 0, veliko možnosti za skrivanje.

Če želimo skriti določene procese, ki tečejo na operacijskem sistemu, potem moramo zamenjati funkcije v jedru ali pa filtrirati rezultate. Ker je rootkit že del jedra, se lahko odločimo in spremenimo funkcije.

KeServiceDescriptorTable je tabela, izvožena iz jedra in ima kazalec (pointer) v SSDT (System Service Dispatch Table), kjer se nahajajo glavni servisi in funkcije, ki jih implementira Ntoskrnl.exe. Ko je rootkit uspešno nameščen, lahko spreminja SSDT, ki nato kaže na funkcije, definirane v rootkitu, namesto na tiste, definirane v Ntoskrnl.exe in Win32k.sys. Dober primer take funkcije je »ZwQuerySystemInformation[12]«, na katero se pogosto zanaša operacijski sistem za seznam vseh trenutnih procesov na sistemu (tudi Taskmgr.exe uporaba to funkcijo).

[12] http://msdn2.microsoft.com/en-us/library/ms725506.aspx

```
NTSTATUS WINAPI ZwQuerySystemInformation(

      SYSTEM_INFORMATION_CLASS SystemInformationClass,
      PVOID SystemInformation,
      ULONG SystemInformationLength,
      PULONG ReturnLength

      );
```

»Runtime patching«

Te in tudi druge podobne »hooking« metode (ki jih je veliko) so do danes že zelo dobro dokumentirane. Tudi programi, ki iščejo zlonamerno kodo, znajo iskati tudi t.i. »hooks«. Zato je zadnje čase vedno bolj popularen t.i. »runtime patching«. »Runtime patching« nas je večina srečala pri »krekanju« programov (npr. WPAKill je tipični primer takega programa) ali pa pri igranju iger. Cilj napadalca opremljenega z rootkitom je spremeniti logiko programa, ki bo sedaj poleg svojega običajnega dela opravljal še kakšno dodatno funkcijo.

V prvem delu moramo odkriti funkcijo, ki jo želimo nadomestiti s svojo. Pri tem si lahko pomagamo s katerim od razhroščevalnikov za jedro (npr. WinDbg[13]). Ko smo našli funcijo, ki jo želimo nadomestiti, jo prepišemo s svojo funkcijo, ki je v tem primeru `far jmp`. Pri tem moramo upoštevati, da niso vse funkcije enako dolge. Npr. `push` funkcija je dolga 1 byte, medtem ko je `far jmp` funkcija dolga 7 bytov in vsaj toliko prostora potrebujemo. V spodnjem primeru[14] imamo navodila (instructions), ki so dolga 9 bytov.

55	8B	EC	53	33	DB	38	5D	24

PUSH	MOV	PUSH	XOR	CMP

Potem, ko smo prepisali navodila sta nam ostala dva byta CMP funkcije, ki pa ju ne moremo kar pustiti, saj za sistem v tej obliki nimata nobenega pomena in bi brez težav lahko prišlo do nestabilnosti sistema (t.i. Blue Screen).

EA	AA	AA	AA	AA	08	00		
FAR JMP								
						CMP		

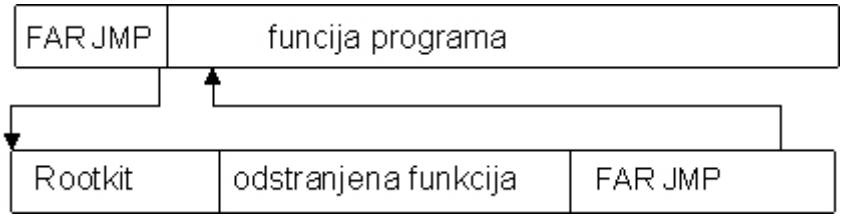
[13] http://www.microsoft.com/whdc/devtools/debugging/default.mspx
[14] Iz knjige Rootkits str. 116

Zato lahko za zadnja dva byta uporabimo funkcijo `nop`, ki je dolga točno en byte (z namenom, saj bo verjetno nekje v prihodnosti uporabljena za »on the fly« nameščanje popravkov).

55	8B	EC	53	33	DB	38	90	90

FAR JMP	NOP	NOP

Ko prepišemo staro funkcijo, je s tem tudi izginila s sistema, na kar moramo paziti. Če želimo, da rootkit ostane neodkrit kar se da dolgo, potem moramo obstoječe funkcije podpirati še naprej. Kar lahko naredimo je, da staro funkcijo prekopiramo v drugi del pomnilnika in jo kličemo po tem, ko se je že izvedla koda rootkita. Tu je priporočljivo, da uporabljamo »non-pageable« del pomnilnika.



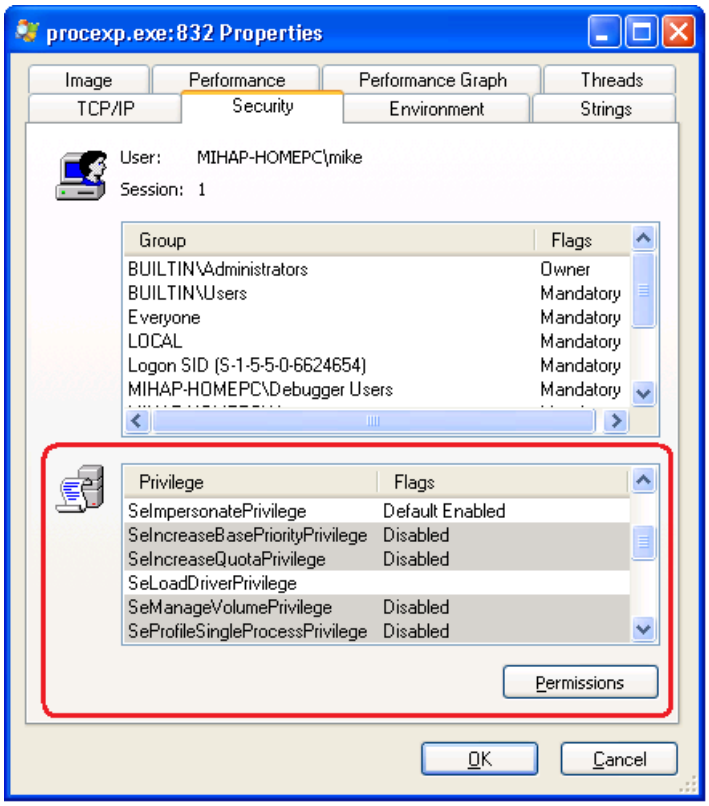
Sistem lahko del pomnilnika, ki je »pageable«, shrani na disk in v nekaterih primerih se lahko pojavijo težave z dostopom do teh informacij prenešenih na trdi disk. V skrajnem primeru bo sistem vrnil »Blue Screen«. Najlažje se temu izognemo tako, da prekopiramo funkcijo v »non-pageable« del pomnilnika.

DKOM (Direct Kernel Object Manipulation)

Kot del jedra lahko z rootkitom spreminjamo tudi žetone (token) obstoječih procesov in niti (threads). V Windows operacijskem sistemu imajo vsi procesi in niti svoj žeton, ki definira pravice, ki jih proces ali nit ima. Nit ima sicer lahko povsem drugačen žeton in s tem pravice kot proces, ki je nit kreiral, vendar je to v praksi zelo redko videti.

Z orodjem »Process Explorer[15]« lahko pogledamo katere pravice ima nek proces dodeljene.

[15] http://www.microsoft.com/technet/sysinternals/utilities/processexplorer.mspx



Obstajajo sicer »uradne« poti, ki nam omogočajo spreminjanje žetonov procesov. V ta namen bi lahko uporabili API klice kot so »OpenProcessToken()«, »AdjustTokenPrivilegies()« in »AdjustTokenGroups()«. Vendar pa v tem primeru potrebujemo pravice kot so »TOKEN_ADJUST_GROUPS« IN »TOKEN_ADJUST_PRIVILEGIES«. Ker je naš rootkit že del jedra, si lahko privoščimo spreminjanje žetona tudi brez zgoraj omenjenih pravic ter mimo zgoraj omenjenih API klicev.

Moramo pa se zavedati, da je spreminjanje žetonov precej zapleteno. Sestavljeni so iz statičnih in variabilnih delov. Variabilni del žetona je tisti, ki nas v tem trenutku najbolj zanima, saj so v njem zapisani SIDi, ki pripadajo žetonu. Ne smemo pa pozabiti tudi na statični del, ki nosi informacije o velikosti žetona, številu definiranih SIDov itn.

Struktura žetona, kot je vidna z razhroščevalnikom jedra (kernel debugger)

```
+0x000 TokenSource      : _TOKEN_SOURCE
+0x010 TokenId          : _LUID
+0x018 AuthenticationId : _LUID
+0x020 ParentTokenId    : _LUID
+0x028 ExpirationTime   : _LARGE_INTEGER
+0x030 TokenLock        : Ptr32 _ERESOURCE
+0x038 AuditPolicy      : _SEP_AUDIT_POLICY
+0x040 ModifiedId       : _LUID
+0x048 SessionId        : Uint4B
+0x04c UserAndGroupCount : Uint4B
+0x050 RestrictedSidCount : Uint4B
+0x054 PrivilegeCount   : Uint4B
```

```
   +0x058 VariableLength   : Uint4B
   +0x05c DynamicCharged   : Uint4B
   +0x060 DynamicAvailable : Uint4B
   +0x064 DefaultOwnerIndex : Uint4B
   +0x068 UserAndGroups    : Ptr32 _SID_AND_ATTRIBUTES
   +0x06c RestrictedSids   : Ptr32 _SID_AND_ATTRIBUTES
   +0x070 PrimaryGroup     : Ptr32 Void
   +0x074 Privileges       : Ptr32 _LUID_AND_ATTRIBUTES
   +0x078 DynamicPart      : Ptr32 Uint4B
   +0x07c DefaultDacl      : Ptr32 _ACL
   +0x080 TokenType        : _TOKEN_TYPE
   +0x084 ImpersonationLevel : _SECURITY_IMPERSONATION_LEVEL
   +0x088 TokenFlags       : Uint4B
   +0x08c TokenInUse       : Uchar
   +0x090 ProxyData        : Ptr32 _SECURITY_TOKEN_PROXY_DATA
   +0x094 AuditData        : Ptr32 _SECURITY_TOKEN_AUDIT_DATA
   +0x098 OriginatingLogonSession : _LUID
   +0x0a0 VariablePart     : Uint4B
```

Žeton Process Explorerja, kot je viden z razhroščevalnikom jedra

```
_TOKEN e56fc030

TS Session ID: 0x1
User: S-1-5-21-220523388-1960408961-1801674531-1009
Groups:
 00 S-1-5-21-220523388-1960408961-1801674531-513
    Attributes - Mandatory Default Enabled
 01 S-1-1-0
    Attributes - Mandatory Default Enabled
 02 S-1-5-21-220523388-1960408961-1801674531-1014
    Attributes - Mandatory Default Enabled
 03 S-1-5-32-544
    Attributes - Mandatory Default Enabled Owner
 04 S-1-5-32-545
    Attributes - Mandatory Default Enabled
 05 S-1-5-14
    Attributes - Mandatory Default Enabled
 06 S-1-5-4
    Attributes - Mandatory Default Enabled
 07 S-1-5-11
    Attributes - Mandatory Default Enabled
 08 S-1-5-5-0-6624654
    Attributes - Mandatory Default Enabled LogonId
 09 S-1-2-0

    Attributes - Mandatory Default Enabled
Primary Group: S-1-5-21-220523388-1960408961-1801674531-513

Privs:
 00 0x000000017 SeChangeNotifyPrivilege           Attributes - Enabled Default
 01 0x000000008 SeSecurityPrivilege               Attributes - Enabled
 02 0x000000011 SeBackupPrivilege                 Attributes - Enabled
 03 0x000000012 SeRestorePrivilege                Attributes -
 04 0x00000000c SeSystemtimePrivilege             Attributes -
 05 0x000000013 SeShutdownPrivilege               Attributes -
 06 0x000000018 SeRemoteShutdownPrivilege         Attributes -
 07 0x000000009 SeTakeOwnershipPrivilege          Attributes -
 08 0x000000014 SeDebugPrivilege                  Attributes - Enabled
 09 0x000000016 SeSystemEnvironmentPrivilege      Attributes -
```

```
 10 0x00000000b SeSystemProfilePrivilege          Attributes -
 11 0x00000000d SeProfileSingleProcessPrivilege   Attributes -
 12 0x00000000e SeIncreaseBasePriorityPrivilege   Attributes -
 13 0x00000000a SeLoadDriverPrivilege             Attributes - Enabled
 14 0x00000000f SeCreatePagefilePrivilege         Attributes -
 15 0x000000005 SeIncreaseQuotaPrivilege          Attributes -
 16 0x000000019 SeUndockPrivilege                 Attributes - Enabled
 17 0x00000001c SeManageVolumePrivilege           Attributes -
 18 0x00000001e Unknown Privilege                 Attributes - Enabled Default
 19 0x00000001d SeImpersonatePrivilege            Attributes - Enabled Default
Authentication ID:         (0,651614)
Impersonation Level:       Anonymous
TokenType:                 Primary
Source: User32             TokenFlags: 0x89 ( Token in use )
Token ID: a785ec3          ParentToken ID: 0
Modified ID:               (0, a786f3c)
RestrictedSidCount: 0      RestrictedSids: 00000000
```

Odločiti se moramo tudi katere privilegije bomo dodali žetonu. Nekaj primerov privilegijev:

- SeDebug
- SeLoadDriverPrivilege
- SeCreateTokenPrivilege

Če si bolj natančno pogledamo zgornjo sliko procesa in njegovih pravic opazimo, da je kar nekaj od teh nastavljenih na »disabled«. To nam lahko močno koristi, saj ne moremo kar preprosto povečevati velikosti žetona in prepisovati pomnilnika, ki direktno sledi žetonu, saj o tem naslovu ničesar ne vemo (lahko tudi, da bi zašli v neveljavne naslove pomnilnika). Zato pa lahko obstoječe »disabled« pravice prepišemo z novimi ter po potrebi dodamo SIDs.

»DLL Injection«

»DLL inejction« se večino časa uporablja v uporabniškem načinu delovanja in ga lahko preprosto izvedemo s spremembo sistemskega registra »`HKLM\Software\Microsoft\Windows NT\CurrentVersion\Windows\AppInt_DLL`« (prisoten na Windows NT, Windows 2000, Windows XP, Windows 2003, Windows Vista), kamor rootkit doda svojo dll datoteko. Preko te dll datoteke, lahko rootkit vpilva na IAT (Import Address table) ali pa na datoteke kernel32.dll in ntdll.dll. Ko neka aplikacija kliče (uporablja) User32.dll, le ta naloži v naslovni prostor aplikacije tudi dll, ki ga je v `AppInt_DLL` kot del registra vpisal rootkit. S tem je naš rootkit dobil polni dostop vsaj do aplikacije.

Drugi možni način za »DLL injection« je uporaba »`CreateRemoteThread`[16]«, kjer se rootkit požene v naslovnem prostoru drugega procesa. S tem rootkit dobi dostop do vseh niti (thread), ki so del istega procesa.

```
HANDLE WINAPI CreateRemoteThread(

      HANDLE hProcess,
      LPSECURITY_ATTRIBUTES lpThreadAttributes,
      SIZE_T dwStackSize,
      LPTHREAD_START_ROUTINE lpStartAddress,
      LPVOID lpParameter,
      DWORD dwCreationFlags,
      LPDWORD lpThreadId

);
```

Uporabna vrednost te metode je opisana nekoliko pozneje.

Operacijski sistemi so sicer vedno bolj prilagojeni na uporabniško delo z omejenim naborom pravic. Če smo bili do nedavnega vajeni poganjati vsakodnevne aplikacije kot lokalni skrbnik, se to precej spreminja z Microsoft Vista in vgrajenim »User Account Control« (UAC). Spremembe v omenjenem operacijskem sistemu vplivajo tako na končne uporabnike kot na razvijalce aplikacij in avtorje rootkitov.

Že v preteklosti so se pojavili rootkiti, ki so dobro delovali tudi z omejenim naborom pravic. Ti rootkiti sicer nimajo nadzora nad celotnim operacijskim sistemov, vseeno pa imajo poln dostop do podatkov (datoteke), do katerih ima dostop uporabnik ter nadzor na aplikacijami, ki jih poganja uporabnik.

Je pa tovrstne rootkite lažje odkriti, saj težje »lažejo« operacijskemu sistemu. Ti rootkiti se sicer lahko skrijejo pred osnovnimi orodji (npr. task manager), vseeno pa ne morejo manipulirati z jedrom.

Za komunikacijo z omrežjem lahko rootkit, ki teče v uporabniškem načinu, uporablja samo TDI (Transport Driver Interface) in nimajo direktnega dostopa do NDIS (Network Driver Interface Specification). Za razliko od NDIS, TDI ne podpira t.i. »raw socket«, komunikacijo z omrežjem pa lahko v tem primeru prepreči preprost osebni požarni zid, HIDS (Host Intrusion Detection Systems) ali NIDS (Network Intrusion Detection Systems).

Če ima rootkit dostop do NDIS[17] (v tem primeru ima rootkit dostop do jedra), potem lahko uporaba »raw sockets«, si npr. izbere MAC in IP naslov ter se lahko izogne osebnim požarnim zidovom, HIDS in NIDS.

Uporabnikom, ki vsakodnevna opravila izvajajo kot lokalni skrbniki (local administrators), ali še slabše, domenski skrbniki (domain adminstrators), ni možno odvzeti

[16] http://msdn2.microsoft.com/en-us/library/ms682437.aspx
[17] Tipični primer rootkita, ki uporablja »hooks« v NDIS je »DeepDoor«

posameznih pravic (kljub temu, da take zahteve pogosto prejemam). Ne glede na poskuse omejevanja (npr. preko lokalne ali skupinske politike) bodo lokalnih skrbniki zmeraj imeli poln dostop do vseh sistemskih virov vključno z jedrom in samo vprašanje časa (in iznajdljivosti uporabnika) je, kdaj tak uporabnik prevzame celoten nadzor nad računalnikom.

Spremembam, ki jih prinaša Windows Vista in UAC, se bodo prilagodili tudi avtorji rootkitov. Ti bodo s časom izgubili enostaven dostop do kernela (s časom bo vedno manj uporabnikov lokalnih skrbnikov) in se bodo posvetili rootkitom, ki delujejo tudi z omejenim naborom pravic.

Omrežje

Že dolgo časa je znano, da osebni požarni zid lahko pomaga pri preprečevanju kraje informacij z računalnika ter celo pred okužbami s črvi (npr. Blaster). Vendar pa se lahko na osebne požarne zidove zanašamo predvsem za prihajajoči (inbound) promet. Odhajajoči promet (outbound) je sicer možno filtrirati, vendar se lahko na to funkcionalnost zanašamo samo toliko časa, dokler uporabljamo aplikacije, ki se »lepo obnašajo«. »Lepo obnašanje« pa ni lastnost, ki bi jo lahko pripisali npr. virusom, črvom, rootkitov, spyware programom itn. Ti programi imajo ogromno možnosti, da zaobidejo osebni požarni zid. Ena od možnosti za je uporaba funkcije »`CreateRemoteThread`«[18], kjer določimo navidezni naslovni prostor (virtual address space) nekega drugega procesa, ki je v našem primeru lahko IE (Internet Explorer), Firefox, Outlook, OutlookExpress ali katera koli druga aplikacija, ki sme vzpostavljati povezave skozi osebni požarni zid. Osebni požarni zid tako komunikacijo prepozna kot legitimno in je ne ustavi.

Tudi (H)IDS in podobni sistemi imajo tu omejeno vrednost, saj zlonamerna koda (virusi, črvi, spyware, rootkiti) vedno bolj pogosto uporabljajo šifrirne mehanizme (SSL/TLS, steganografijo[19], ...) za svoje prikrito delovanje in krajo informacij. Kljub temu, da obstaja programska oprema[20], ki IDS in IPS sistemom omogoča analizo in vpogled v SSL/TLS šifriran promet, se le-ta uporablja izredno redko. Tako IDS/IPS sistemi ne ločujejo obiska legitimne strani (npr. spletne banke) ter pošiljanje ukradenih informacij na spletne strani kriminalcev preko SSL/TLS šifriranih povezav. Pri nadzoru šifriranih povezav ne smemo pozabiti tudi na (slovensko) zakonodajo, kjer lahko s spremljanjem vsebine šifriranega prometa hitro kršimo zakone (npr. zakon o varstvu osebnih podatkov[21]).

Zlonamerna koda lahko tudi oponaša že znane vrste napadov (npr. črv Blaster) ter s tem prelisiči predvsem sistemskega skrbnika, ki morda ne bo pravočasno reagiral na opozorila IDS/IPS sistema.

[18] http://msdn2.microsoft.com/en-us/library/ms682437.aspx
[19] http://en.wikipedia.org/wiki/Steganography
[20] http://ettercap.sourceforge.net/
[21] http://www.ip-rs.si/index.php?id=425

Avtor rootkita lahko implementira dodatno programsko logiko, ki upošteva količino prometa na omrežju. Tako prilagojen rootkit izvaja aktivnosti na omrežju le v času večje obremenitve, saj tako lažje skrije svoje delovanje. Avtor v tem primeru računa na nepazljivost sistemskega skrbnika pri pregledovanju log zapisov na aktivni opremi (npr. požarni zid), ki v veliki količni prometa ne bo opazil delovanje rootkita. Avtorji rootkita se pogosto zanašajo tudi na to, da bo rootkit na okuženem sistemu dolgo časa neopazen. Zato si lahko za krajo informacij vzamejo čas in jih preko omrežja prenašajo počasi, s čimer se izognejo morebitni pozornosti sistemskega skrbnika.

Rootkiti, specializirani za delovanje na omrežju, imajo lahko svoj MAC (Media Access Control) in IP naslov. Tak okužen sistem lahko v omrežju predstavlja še dodatne izzive. Četudi sistemski skrbnik zazna nenavaden ali celo nevaren promet na omrežju, je lahko identifikacija okuženega sistema težavna in večje kot je število sistemov na omrežju, bolj kompleksna je naloga. Najlažje je tako okužen sistem odkriti v okoljih, kjer je vsak od računalnikov priključen direktno na pametno mrežno stikalo, ki omogoča iskanje, na katerih vratih se nahaja določeni MAC naslov. Ni nujno, da se tu iskanje zaključi. Standardna orodja in ukazi kot je npr. »`ipconfig /all`« ne bodo razkrila MAC ali IP naslova rootkita in možno je, da bo sistemski skrbnik spregledal okužen sistem. Lahko je na omenjena vrata priključenih večje število sistemov. Če ne gre za fizični priključek pa lahko na sistemu teče večje število virtualnih okolij in katero koli od teh okolij je potencialno okuženo[22].

Del izvorne kode iz rootkita rk_044[23], kjer definiramo lasten MAC naslov.

```
__int64 our_mac = 0xADDEEFBEADDE; //deadbeefdead
```

Omenili smo že popularne brskalnike, ki jih je možno zelo učinkovito in preprosto izkoriščati za krajo podatkov. Uporabniki pogosto nadgradijo svoje spletne brskalnike z različnimi dodatki (plug-ini), ki pa imajo lahko poleg koristne tudi zlonamerno vlogo. Tako lahko dodatki prisluškujejo zaupnim informacijam (npr. številke kreditnih karic, gesla za elektronsko bančništvo, ...) ko jih uporabnik vnaša v spetno formo, še preden jih brskalnik zašifrirane pošlje preko SSL/TLS povezave.

Pogosto je dostop do spletnih bank zaščiten z močnimi mehanizmi preverjanja identitete (dvofaktorska prijava), ki pa nas v zgoraj omenjenem primeru ne varuje. Na internetu obstaja kar nekaj učinkovitih napadov na sisteme, ki uporabljajo enkratna gesla (tipični primer je virus W32.Grams[24]) ali pametne kartice.

[22] Avtor članka ve za večje slovensko poslovno omrežje kjer je nekaj virtualnih računalnikov povzročilo zaustavitev celotnega omrežja. Ti virtualni računalniki so se uporabljali precej redko in niso imeli nameščenih ustrenih popravkov, ki bi preprečevali okužbe s črvi. Ko so bili virtualni sistemi okuženi so generirali toliko prometa, da se je omrežje ustavilo.

[23] Celotna izvorna koda rootkita rk_44 je na voljo na spletnem naslovu http://www.rootkit.com/vault/hoglund/rk_044.zip

[24] http://www.oregonhipaaforum.org/Files/2005%20Attack%20Trends%20%20Analysis--CIS-MLfinal.pdf

Zlonamerna koda (rootkit, ki je lahko nameščena kot plug-in v spletnem brskalniku) počaka, da se uporabnik uspešno prijavi na npr. račun spletne banke (lahko uporabi enkratni sistem gesel ali pametno kartico). Ko je uporabnik uspešno prijavljen, ima zlonamerna koda (ki je v teh primerih običajno prilagojena tako, da napade točno določeno banko) poln dostop do bančnega računa in lahko izvaja transakcije in okuženemu uporabniku izprazni bančni račun. Najbolj učinkovita metoda zaščite pred tovrstnimi napadi je avtentikacija vsake transakcije in ne samo začetnega vstopa v spletne banke.

Preglede lastnosti rootkitov glede na način delovanja:

Jedrni način delovanja

Prednosti	**Slabosti**
Se enostavno skrijejo	Pisanje jedrnih rootkitov je izredno zahtevno
Jih je težko odkriti	Slabo napisan rootkit lahko vpliva na stavilnost sistema
Imajo dostop do celotnega sistema	

Rootkiti, ki se lahko skrivajo v stojni opremi (BIOS, ...)

Prednosti	**Slabosti**
Jih je težko odkriti	Pisanje rootkitov, ki se lahko skrivajo v strojni opremi je izredno zahtevno
Niso odvisni od operacijskega sistema	

Uporabniški način delovanja

Prednosti	**Slabosti**
Razvoj je precej bolj enostaven kot pri rootkitih, ki delujejo v jedru ali strojni opremi	Se težje skrijejo pred orodji za odkrivanje zlonamerne kode
	Okužijo lahko samo uporabnika, ki je zlonamerno kodo pognal
	Imajo dostop samo do informacij, do katerih ima dostop uporabnik, ki je zlonamerno kodo pognal

Odkrivanje RootKitov

Najti je možno kar nekaj orodij za odkrivanje rootkitov npr. BlackLight[25] in RKR (RootkitRevealer)[26], če omenimo dva. Žal pa je tu podobno kot pri virusih, kjer imajo avtorji

[25] http://www.f-secure.com/blacklight/blacklight.html
[26] http://www.microsoft.com/technet/sysinternals/utilities/rootkitrevealer.mspx

zlonamerne kode prednost pred avtorji zaščite. Avtorji zlonamerne kode lahko zelo hitro prilagajajo svoje programe, da se bolje skrivajo pred programi za odkrivanje rootkitov.

Rootkitu večino časa ni potrebno biti zapisan v registru operacijskega sistema. Če je rootkit napisan kot servis ali gonilnik, bo v primeru ponovnega zagona računalnika dobil obvestilo od operacijskega sistema, kjer lahko sedaj izvede »`OnShutdown`[27]« oz. kateri drugi podoben event in se šele sedaj zapiše v npr. »`RunOnce`[28]« del sistemskega registra (če ne implementira funkcije skrivanja). Ko se računalnik ponovno zažene prebere »`RunOnce`« del registra, ki pobriše vpisan ukaz in požene npr. rootkit oz. kakšno drugi zlonamerno kodo. Na ta način se lahko rootkit izogne odkritju z npr. rednim skeniranjem operacijskega sistema.

Rootkitu niti ni potrebno biti naložen v pomnilniku. Dovolj je, da se zažene ob zagonu računalnika, spremeni delovanje nekega programa ter se nato odstrani iz spomina. Dokler bo spremenjeni program deloval dalje, bo izvajal funkcije, ki mu jih je vsilil rootkit. Tipični primer takega delovanja je opisan zgoraj pod »Runtime patching«.

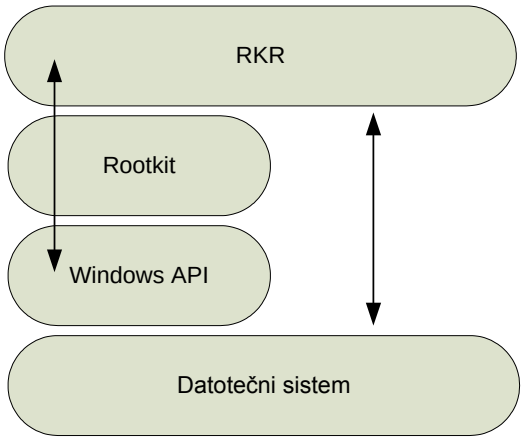
Nekateri programi za odkrivanje rootkitov poskušajo zaznati take spremembe funkcionalnosti v programih vendar pogosto pogledajo samo prvih 20 byte-ov procesa, ker je tu zaradi kompleksnosti najlažje spremeniti delovanje programa. Vse, kar mora rootkit storiti je, da program spremeni nekje drugje (kar sicer močno poveča kompleksnost rootkita, mu pa to zagotovi daljši obstoj na sistemu).

Eden od programov, ki deluje predvsem preventivno je Tripwire[29]. Tripwire lahko obvesti sistemskega skrbnika kadar se na sistemu zgodi nepooblaščena sprememba sistemskega registra ali sistemskih datotek. Je pa tudi Tripwire, pa tudi veliko drugih programov nemočen kadar gre za rootkite, ki »živijo« samo v delovnem pomnilniku (»Runtime patching«).

Kadar nameščamo tovrstno programsko opremo se moramo zavedati, da je učinkovita le če sistem še ni okužen.

RKR poskuša rootkite odkriti tako, da najprej preveri informacije (katere datoteke se nahajajo na trdem disku, informacije o sistemskem registru, ...), ki jih dobi od operacijskega sistema preko API klicev. Nato RKR pregleda trdi disk ter register mimo API klicev (»raw« format datotečnega sistema ter registra). Če se seznama datotek in registrov razlikujeta, smo operacijski sistem zalotili pri »laži« in velika verjetnost je, da se okužen z rootkitom. Razlike v seznamih nam lahko pomagajo identificirati rootkit, ki je nameščen na sistemu.

[27] http://msdn2.microsoft.com/en-us/library/system.serviceprocess.servicebase.onshutdown.aspx
[28] http://msdn2.microsoft.com/en-us/library/aa376977.aspx
[29] www.tripwire.org



Skica1: Delovanja RKR

Nekateri rootkiti so se že prilagodili na prve različice RKRja in ko zaznajo, da je na sistemu pognan RKR prenehajo s skrivanjem informacij, ki jih RKR išče. Novejše različice RKRja so odporne na te prilagoditve rootkitov.

Odstranjevanje rootkitov

Za najbolj učinkovito metodo odstranjevanja rootkitov še zmeraj velja ponovna namestitev operacijskega sistema (priporočeno tudi s strani Microsofta[30]). Obstajajo pa rootkiti, ki so odporni tudi na ponovno nameščanje, saj se lahko skrivajo v strojni opremi (npr. BIOS kar je lahko tudi BIOS na PCI napravah) in so neodvisni od operacijskega sistema.

Glavni izziv pri odstranjevanju rootkitov je vprašanje, ali smo odstranili vse rootkite z okuženega sistema ali pa smo morda zaznali in odstranili samo enega od nekaj rootkitov, ki tečejo na sistemu.

Na voljo imamo tudi t.i. »offline« analizo trdega diska, ki pa ni brez svojih izzivov. Tudi v tem primeru se rootkiti lahko še zmeraj skrivajo v npr. ADS (Alternative Data Streams) v MBR (Master Boot Record), v BIOSu...

Večji del kode rootkita je lahko na trdem disku zašifrirana. Ko se okuženi sistem zažene, prebere BIOS vključno z rootkitom, ga naloži v pomnilnik, kjer ima sedaj rootkit enostaven dostop do svoje kode (samo rootkit ima zasebni ključ ali algoritem preko katerega lahko dešifrira shranjene informacije). Tako šifriranje pa lahko oteži analizo diska, saj ne vemo, komu datoteka pripada in v kakšen namen se uporablja.

Če izvajamo »offline« analizo trdega diska, je smiselno preveriti digitalne podpise sistemskih datotek. Kadar datoteke niso digitalno podpisane, lahko uporabimo zgoščevalne

[30] http://www.microsoft.com/technet/community/columns/sectip/st1005.mspx

fukcije (hash funkcije) ter vrednosti primerjamo s sistemom, za katerega smo prepričani, da ni okužen. Pri tem moramo paziti, da imata oba sistema, ki ju primerjamo, nameščene iste servisne popravke (service pack) ter popravke.[31]

Zaščita

Najbolj učinkovita zaščita pred rootkiti je vsekakor preventiva, ki vključuje vsakodnevno uporabo računalnika s pravicami omejenega uporabnika. S tem lahko predvsem preprečimo namestitev rootkitov, ki tečejo kot del jedra, ne pa tudi rootkitov, ki tečejo s pravicami uporabnika in lahko še zmeraj povzročijo veliko škode (npr. kraja zaupnih dokumentov, do katerih ima dostop uporabnik, ki se je okužil). Odvisno od načina »okužbe« z rootkitom so lahko protivirusni in drugi varnostni programi nemočni. Spomnimo se npr. virusov Blaster in Code Red, ki sta uspešno, preko varnostne luknje v operacijskem sistemu, lahko zaobšla tudi protivirusno zaščito.

Verjetno bomo nekje v prihodnjih letih pričeli bolj nadzirati komunikacijo v omrežjih in bomo omejevati dostope do strani, ki so zaščitene s SSL/TLS oz. bomo dovoljevali komunikacijo preko šifriranih kanalov izključno s stranmi, ki jim zaupamo (npr. spletne banke...).

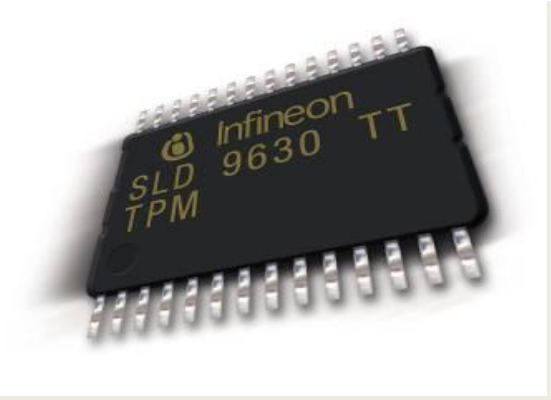
V 64 bitni različici Windows Vista ter v Microsoft strežniškem operacijskem sistemu zaenkrat znanega pod kodnim imenom Longhorn je vgrajena dodatna zaščita za jedro. Ta dva operacijska sistema, za razliko od 32 bitnih različic, ne naložita gonilnikov, ki niso digitalno podpisani oz. če je digitalni podpis neveljaven (npr. zaradi nepooblaščene spremembe datoteke). To zaščito je sicer možno zaobiti, če ima napadalec fizični dostop do računalnika ter uporabniško ime in geslo sistemskega skrbnika, saj mora sistem pognati v varnem načinu delovanja (Safe Mode) oz. mora na računalnik priključiti jedrni razhroščevalnih (Kernel Debugger).

Pred fizičnimi napadi so trenutno najbolj izpostavljeni prenosniki, ki jih uporabniki nosijo iz nekega varnega okolja v okolja, o katerih ne vemo ničesar. Prepogosto pozabimo tudi na fizično varnost strežnikov, ki so nahajajo na oddaljenih lokacijah, kjer pogosto ni primerno poskrbljeno za varnost računalniše opreme. Najbolj učinkovita zaščita samega sistema (tudi kot zaščita pred BIOS rootkiti) v teh primerih izhaja iz same strojne opreme – TPM (Trusted Platform Module), ki pa še ni močno razširjena.

[31] Orodje, ki ga lahko uporabimo v ta namen je brezplačno na voljo na Microsoft spletni strani http://support.microsoft.com/kb/841290

TPM

TPM deluje predvsem kurativno, saj ne more preprečiti same spremembe BIOSa ali sistemskih datotek. Bo pa sistem, z uporabo TPM opazil vse nepooblaščene spremembe. V primeru, ko pride do nepooblaščenih sprememb se operacijski sistem ne naloži, s čimer je preprečen tudi dostop do informacij na njem.



Windows Vista lahko (ni pa pogoj) TPM uporablja skupaj s t.i. tehnologijo »BitLocker«, ki omogoča šifriranje celotnega zagonskega nosilca (boot volume) (nosilec (volume), na katerem se nahaja Windows mapa). Glavna vloga TPM pa je hranjenje zasebnih ključev, ki se nato uporabljajo za različne šifrirne operacije. TPM-ju bi lahko rekli tudi pametna kartica za računalnike. Tako kot uporabniki na pametnih karticah shranjujemo zasebne ključe, ki nam omogočajo privilegirane operacije (npr. dostop do spletne banke), računalnik uporablja TPM za shranjevanje zasebnih ključev, ki mu omogočajo privilegirane operacije (npr. zagon operacijskega sistema). Zasebni ključi, ki so shranjeni na TPM čipu (ali pametni kartici), tega nikoli več ne zapustijo.

Ko lastnik sistema prvič prevzame lastništvo (faza se po angleško imenuje »taking ownership«) nad TPM se na njem zgenerira RSA 2048-bit par ključev. Ta ključ se imenuje SRK (Storage Root Key). Ko v Windows Vista vključimo BitLocker, se kreira 128-bit AES FVEK (Full Volume Encryption Key), ki je dodeljen določenemu nosilcu. FVEK je nato zašifriran z VMK (Volume Master Key) in shranjen kot metapodatek (metadata) na posebnem delu nosilca. Ob vklopu BitLocker-ja imamo na izbiro več možnosti za zaščito VMK, med drugim tudi tako, da ga zašifriramo z zasebnim ključem shranjenim v TPM, ki pa pri tem upošteva tudi informacije shranjene v PCR (razloženo spodaj). Uporabljamo lahko tudi več kombinacij zaščite VMK. Uporabimo lahko npr. kombinacijo TPM in USB ključ ali pa TPM in PIN, ki ga moramo vtipkati ob zagonu sistema (od 4 do 20 numeričnih znakov).

TPM čip ima 24 (od 0 – 23) t.i. »Platform Configuration Registers« oz. PCR[32], vendar se v praksi uporablja samo prvih 16 (od 0 – 15). Ob vključitvi BitLocker-ja ter ob vsaki pooblaščeni spremembi, Windows Vista izračuna SHA-1 izvleček (SHA-1 hash) (informacija se

[32] http://msdn2.microsoft.com/en-us/library/aa376469.aspx

v ang. literaturi imenuje »measurement«), ki predstavlja strojno ali programsko opremo sistema ter to vrednost shrani v namenjen PCR (npr. informacije o BIOS-u gredo v PCR[0]).

PCR[23]	**...**
...	**...**
PCR[15]	Defined for use by the static operating system
PCR[14]	Defined for use by the static operating system
PCR[13]	Defined for use by the static operating system
PCR[12]	Defined for use by the static operating system
PCR[11]	**BitLocker Access Control**
PCR[10]	**BootManager**
PCR[9]	**NTFS Boot Block**
PCR[8]	**NTFS Boot Sector**
PCR[7]	Computer Manufacturer-Specific
PCR[6]	State Transition and Wake Events
PCR[5]	Master Boot Record (MBR) Partition Table
PCR[4]	**Master Boot Record (MBR) Code**
PCR[3]	Option ROM Configuration and Data
PCR[2]	**Option ROM Code**
PCR[1]	Platform and Motherboard Configuration and Data
PCR[0]	**Core Root of Trust of Measurement (CRTM), BIOS, and Platform Extensions**

Ob vsakem zagonu računalnika, sistem sam izračuna SHA-1 meritve za CRTM (Core Root of Trust of Measurement) ter BIOS ter to shrani v ustrezen PCR. Nato BIOS izračuna SHA-1 za naslednjo komponento v zagonskem procesu – MBR (Master Boot Record) in jo shrani v ustrezen PCR. Proces se nadaljuje vse do nalagalnika operacijskega sistema (operating system loader). Od tu dalje, je vsak proces nalaganja operacijskega sistema odgovoren za izračin SHA-1 naslednje komponente ter shranjevanje informacij v ustrezne PCR. Ko proces nalaganja sistema pride do »Boot Manager«, ta prebere zašifriran VMK in »prosi« TPM, da ga dešifrira. To pa bo TPM lahko naredil samo pod pogojem, da so vse trenutno vrednosti PCR, ki jih je sistem izračunal ob zagonu, enake tistim ob prvotnem kreiranju VMK.

Privzeto sistem preveri PCR vrednosti 0, 2, 4, 8, 9, 10 in 11 (v zgornji tabeli označeno z **debelim** tiskom). Po želji, lahko v omenjenem operacijskem sistemu, preverjanje posameznih PCR vključimo ali izključimo.

Zaključek

Če smo se do nedavnega bali napadov na internetu, se bomo verjetno v prihodnosti bali napadov v vsakdanjem življenju. Če pomislimo na vse naprave, ki jih priklapljamo na

internet (mikrovalovne pečice[33], hladilniki[34], avtomobili[35]) je samo vprašanje časa, kdaj se bodo napadi preselili tudi na te in druge podobne naprave, ki jih bomo priklopili na internet. Morda niti ne bo potrebno iskati ranljivosti v napravah/programski opremi, ampak se bomo lahko še naprej zanašali na človeški faktor[36] in morebitne stare napake[37]

Kriminalne združbe ponujajo tudi več kot 50.000 USD[38] za neobjavljene ranljivosti v operacijskih sistemih. Istočasno je možno za nekaj 100 USD na internetu kupiti naročniku prirejene trojanske konje, ki bodo zbirali in posredovali specifične informacije po željah naročnika. Za vse tiste, ki denarja (še) nimajo, a vseeno »potrebujejo« trojanskega konja, rootkit ali drugo obliko zlonamerne kode, obstaja kar nekaj spletnih strani, kjer je možno brezplačno dobiti orodja za izdelavo le teh. Ena od takih strani je Metasploit Framework[39], ki je sicer primarno namenjena za izdelavo orodij za testiranje varnosti v informacijskih sistemih.

V prihodnosti lahko pričakujemo vedno več zlonamerne kode (Malware), ki bo uporabljala tehnike rootkitov za svoje prikrito delovanje. Rootkite, ki delujejo v uporabniškem načinu je relativno preprosto pisati, imajo pa dostop do vseh uporabniških informacij. Informacije, ki bodo zanimale »lastnike« tovrstnih rootov so poleg očitnih gesel tudi številke kreditnih kartic, ki jih lahko naknadno prodajo za nekaj 100 USD. Številke lahko rootkit dobi ob vpisovanju v spletno stran saj nas SSL/TLS ščiti šele, ko so podatki na poti od našega računalnika do računalnika na drugi strani mreže. SSL/TLS ne zagotavlja varnosti na samih računalnikih, ki si informacije izmenjujeta.

[33] http://www.lginternetfamily.co.uk/micro.asp
[34] http://www.lginternetfamily.co.uk/fridge.asp
[35] http://www.autoblog.com/2007/03/17/you-are-big-brother-control-and-track-your-car-from-the-net/ oz. http://tinyurl.com/29hwcc
[36] Ex-contractor sentenced for sabotaging Navy subs http://content.hamptonroads.com/story.cfm?story=122352&ran=199274
[37] CERT® Advisory CA-2003-21 GNU Project FTP Server Compromise http://www.cert.org/advisories/CA-2003-21.htmls
[38] http://www.eweek.com/article2/0,1895,2073611,00.asp
[39] http://framework-mirrors.metasploit.com/msf/

Pogosto lahko tudi v slovenskih medijih beremo o bolj ali manj uspešnih poskusih vdorov v informacijske sisteme ali krajo občutljivih informacij (npr. spletne banke, ponudniki internet storitev, ...). Izjave odgovornih ob tovrstnih dogodkih v obliki »v naše omrežje ni možno vdreti« ter »v naše omrežje ni še nihče vdrl« me vedno znova presenečajo, saj jih smatram za neodgovorna dejanja.

Ob tovrstnih izjavah se preprosto moramo vprašati, kako dokažemo, da v omrežje še ni bilo vdrto ter kako izmerimo kako varno je naše omrežje? Ali je to, da nismo »opazili« ničesar »nenavadnega«[40] dovolj, ali je to le posledica pomanjkanja časa/neznanja? Kako veste, da nekdo že nekaj let nima dostopa do vseh vaših ključnih informacij, vendar meni, da še ni pravi čas, da vašemu omrežja zada kritični udarec?

Obstaja tudi nekaj revizorjev informacijskih sistemov, ki na koncu, ko si že skoraj oddahneš in komaj čakaš da bo konec, vprašajo – »Can you prove it«?

Reference

Greg Hoglund and James Butler, *Rootkits*, Addison Wesley, 2005

http://www.rootkit.com

Mark E. Russinovich and David A. Solomon, *Microsoft Windows Internals, Fourth Edition: Microsoft Windows Server(TM) 2003, Windows XP, and Windows 2000*, Microsoft Press, 2004

Morebitne spremembe in popravki dokumenta bodo objavljeni na spletni strani http://www.krneki.net/rootkits

[40] http://www.infoworld.com/archives/emailPrint.jsp?R=printThis&A=/article/07/03/29/HNtjxfiling_1.html