UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Žiga Elsner, Primož Bajželj, Uroš Kastelic, Željko Plesac, Jan Varljen

FORENZIČNA ANALIZA OPERACIJSKEGA SISTEMA LINUX

Seminarska naloga

Mentor: prof. dr. Andrej Brodnik

Ljubljana, 2012

Contents

Povzetek:

Težko bi si pred 15 leti in več zamislili, da bodo računalniki nekoč osrednji cilj forenzične preiskave. V drugem desetletju 21. stoletja, ko pa je človek postal zelo odvisen od računalnikov se to zdi kot nekaj nujno potrebnega. V tem članku bo opisanih nekaj osnovnih principov analize računalniške forenzike z operacijskim sistemom Linux.

Ključne besede: forenzična analiza, preiskava diskov, preiskava omrežij, preiskava brskalnikov

Abstract:

About 15 years ago it would be hard to imagine that computers might be the main focus of criminal investigation. However, now in second decade of 21st century this seems to be necessary as we have become quite dependent on computers. In this article few basic principles of computer forensics analysis with Linux operating system are going to be presented.

Key words: forensic analysis, disc investigation, network investigation, browsers investigation

Uvod v forenzično analizo

Znanost forenzične analize je strokovno orientirana za zbiranje in analizo dokazov. Tehnologija, ki se pri tem uporablja pa obsega računalnike ter programsko opremo. Kraj zločina obsega računalnik, omrežje ter naprave na katere je priklopljen.

Delo forenzičnega preiskovalca, je da kar najbolje zbere in preiskuje izvore dokazov kot so diski, zgodovine dnevnikov, prenosne medije in drugih. Pri tem je zelo pomembno, da se ohrani čimveč podatkov v originalni obliki ter, da se kar najbolje obnovi dogodke, ki so se pojavili ob določenem kaznivem dejanju.

Vsako dejanje, ki se zgodi je lahko drugačno in v enem primeru lahko obsega zgolj posamezen računalnik medtem, ko v drugem lahko obsega cele računalniške sisteme z ogromno količino podatkov, ki jih spremlja zakompliciran obseg dogodkov.

Preiskovanje se lahko v začetku velikokrat izkaže kot iskanje igle v senu, saj je lahko veliko stvari skritih. Prav tako velja, da ne obstaja nek recept, ki korak za korakom pove kaj je potrebno narediti, ampak se je situaciji potrebno sprotno prilagajati.

Pomembno je, da pri vsaki preiskavi skrbno popisujemo vsak korak naše preiskave - na kakšen način smo pridobili dokaze ter katere programe in tehnike smo pri tem uporabili. To je pomembno, da lahko tretja oseba ponovi celotni postopek še enkrat in pridobi enake rezultate. Prav tako nam to lahko v kasnejšem obdobju pomaga, da se spomnimo ter postopek, če je le to potrebno, zagovarjamo na sodišču. Poleg tega pa nam lahko te zapiski pomagajo ob podobnih primerih, kar lahko znatno zmanjša stroške preiskave primera in sojenja prihodnjega računalniškega kaznivega dejanja.

Za forenzično analizo je pomembno, da si za to pripravimo temu namenjen računalniški sistem s katerim bomo čimbolj učinkovito prišli do želenih rezultatov. Le ta naj bi med drugim omogočal analizo diskov, preiskovanje omrežij, brskalnikov, zgodovine in drugo. Primer takega sistema bi lahko bil sestavljen iz sledečih komponent:

- računalnik s hitrim procesorjem,
- matično ploščo z vsaj tremi IDE kontrolerji za trde diske, cd in dvd enote,
- podporo za prenosne medije kot so npr. USB ključki,
- vsaj dva velika trda diska, katera imata dovolj prostora za operacijski sistem, forenzična orodja ter za kopiranje particij ter izbrisanih datotek iz dokaznega računalnika,
- SCSI kartico, ki omogoča priključitev diskov, magnetnih trakov, skenerjev, printerjev,
- magnetne trakove, ki omogočajo shranjevanje velikih particij,
- sistem mora imeti celotno podporo za omrežje na katerem pa ni zagnana nobena omrežna storitev razen SSH (ki se uporablja za prenos datotek in zavarovan oddaljen mrežni dostop),
- operacijski sistem Linux, kjer je ena boljših izbir Red Hat Linux - Shrike (Linux izberemo, ker ima precej boljšo podporo glede na druge operacijske sisteme).

Zelo priročen je seveda tudi prenosni računalnik, ki omogoča, da "forenzični laboratorij" prinesemo kar na mesto zločina.

Priprava na analizo

Pred začetkom analize sistema je potrebno slediti istim osnovnim korakom za pripravo sistema:

- V nekaterih primerih je pred razstavljanjem in premikanjem računalniški sistem potrebno slikati. To je v nekaterih primerih potrebno že v osnovi, zaradi slik kraja zločina. Dodatno pa je to potrebno zaradi dokumentiranja komponent sistema, kar nam kasneje omogoča vrnitev sistema v prvotno stanje.
- Potrebno je pisati dnevnik preiskave v katerega se čimbolj podrobno zapisuje celoten potek dela. Le ta vključuje datum začetka preiskave ter začetek in konec določene aktivnosti. Iz varnostnih razlogov je zelo pomembno tudi beleženje morebitnih prekinitev med aktivnostmi.
  Tak pristop služi sprotnemu ustvarjanju poročila, ki je bolj konsistentno in detajlno kot, če bi ga spisali na koncu. Taki podrobni zapiski bodo občutno boljše pripomogli k predstavi celotnega zločina, kot pa samo velika količina dokazov, ki se naberejo že pri najmanjših računalniških zločinih.
- Preden zaključimo sistem je priporočljivo zbrati osnovne informacije o disk. Pod to spada zgradba datotečnega sistema (*/etc/fstab*), katerega prikaz lahko vidimo na Slika 1, ime gostitelja in IP naslove iz (*/etc/hosts*), ki so prikazani na Slika 2, naprave (*/var/log/dmesg*) in sistemska sporočila (*/var/log/syslog*). To lahko najlažje naredimo s tar arhivom kot je prikazano na Slika 3.

```
# /etc/fstab: static file system information.
#
# Use 'blkid -o value -s UUID' to print the universally unique identifier
# for a device; this may be used with UUID= as a more robust way to name
# devices that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    nodev,noexec,nosuid 0       0
# / was on /dev/sda1 during installation
UUID=f9987346-16aa-47aa-aa61-eaf5c495d25e /               ext4    errors=remount-ro 0       1
# swap was on /dev/sda5 during installation
UUID=07be75c3-6a5f-426a-9b31-bfe85f0e64aa none            swap    sw              0       0
/dev/fd0        /media/floppy0  auto    rw,user,noauto,exec,utf8 0       0
```

Slika 1: Prikaz vsebine datoteke /etc/fstab

```
127.0.0.1       localhost
127.0.1.1       ubuntu

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Slika 2: Prikaz vsebine datoteke /etc/hosts

```
tar -cvzPf system_info /etc/hosts /etc/fstab /var/log/dmesg /var/log/syslog
```

Slika 3: Prikaz ukaza za varnostno kopiranje pomembnih datotek

- V kolikor je možno je nujno potrebno narediti kopijo celotnega diska ter delati na kopiji in ne na originalu! Če delamo na originalu obstaja velika možnost, da že zaradi majhne napake

uničimo ali poškodujemo dokazno gradivo. Original moramo hraniti na varnem mestu, kjer ga ni možno uničiti ali spremeniti.

- Ob začetku je potrebno priklopiti disk na prosta IDE vrata in zagnati sistem. Ob tem moramo biti zelo previdni, da ne poškodujemo diska. Če nimamo prostih IDE vmesnikov potem izklopimo CD-ROM enoto iz IDE vmesnika in nanj priključimo ta disk. V BIOSu je morda ob tem potrebno tudi preklopiti avtomatsko detekcijo tipa diska.
- Zatem moramo identificirati particije na disku z uporabo ukaza **fdisk**. Pri tem ukaza ne smemo uporabljati v interaktivnem načinu, saj s tem lahko spremenimo particijsko tabelo. Prikaz uporabe ukaza oz. identifikacijo particij lahko vidmo na Slika 4. Pri tem lahko opazimo lastnosti diska ter npr. predvidevamo, kje se nahaja Linux sistem.

```
george@ubuntu:~/faks/RF$ sudo fdisk -l /dev/sda

Disk /dev/sda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000d7972

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1        2481    19921920   83  Linux
/dev/sda2            2481        2611     1046529    5  Extended
/dev/sda5            2481_       2611     1046528   82  Linux swap / Solaris
```

Slika 4: Prikaz particijske tabele

- Zaradi varnostnih razlogov moramo generirati MD5 vsoto vsake particije, dobiti bitno sliko diska in jo preveriti s prej dobljenimi vsotami (uporaba **mt** in **dd** orodja). Če se vsote ne ujemajo pomeni, da se je lahko pokvaril le en bit, kar se lahko pojavi npr. zaradi slabih sektorjev ali napake pri samem kopiranju.
- Zatem je potrebno pripeti particijo za katero menimo, da je sistemska particija, ki pa je ne smemo spreminjati. Particijo pripnemo z ukazom **mount -r /dev/sda /mnt** in pri tem uporabimo samo bralni način. Iz pripete particije izpišemo vse datoteke z ukazom **ls -lat /mnt**, kjer nato lahko tudi ugotovimo ali gre res za sistemsko particijo. Z pregledom */mnt/etc/passwd* si lahko izpišemo uporabniške račune, ki so bili ustvarjeni do sedaj. Pri tem moramo biti pozorni na sumljiva imena. Te zapise si shranimo zato, da se kadarkoli kasneje lahko vrnemo nazaj in začnemo preverjati sledi določenega uporabnika. Iz teh zapisov lahko naredimo različne predpostavke o znanju vsiljivca ali lastnika računalnika ter pri tem pazimo, saj so lahko bili le ti tudi načrtno uporabljeni kot diverzija za pravo zlorabo.
- Poskušati moramo zgraditi pomembne dogodke jih povezati z določenimi aktivnostmi ter slediti do njihovega izvora. Po možnosti poskušamo tudi ponoviti napadalčeve korake in s tem odkriti njegov pravi namen ter ugotoviti s katerimi orodji in sistemi je izvedel napad.

Od tu naprej pa se lahko začne forenzična analiza preiskave sistema z standardnimi UNIX orodji.

Forenzično preiskovanje diska

Ko smo prejeli disk v roke (ali sliko diska) ga lahko začnemo forenzično preiskovati. Prvi korak je priklopitev diska v načinu samo za branje. To naredimo z ukazom

mount –r *medij direktorij*

Zastavica –r pomeni, da priklopimo disk v *read-only* načinu (lahko uporabimo tudi –o ro), *medij* je absolutna pot do diska katerega smo priklopili ter *direktorij* mesto na katerega bomo priklopili disk. Na primer, če nam se korenski datotečni sistem nahaja v /dev/hdd2 in ga želimo priklopit na /mnt, potem ukaz izgleda:

mount –r /dev/hdd2 /mnt

Za nadaljnje delo je dobro podrobno poznavanje Linux operacijskega sistema in njegovih posebnosti.

Pridobivanje zbrisanih podatkov

V okolju Linux ne obstaja *file slack,* saj se ob ustvarjanju nove datoteke preostali del sektorja napolni z ničlami ter označi kot *nealociran*. Eden od načinov, kako lahko pridemo do izbrisanih podatkov je da poiščemo vse izbrisane *inode* in z njimi pridobimo izgubljene podatke.

```
examiner1% ils -f linux-ext2 /e1/case2/ext2-bitstream.dd | more
class|host|device|start_time
ils|case|ext2-bitstream.dd|1054082181
st_ino|st_alloc|st_uid|st_gid|st_mtime|st_atime|st_ctime|st_dtime|st_mode|st_nli
nk|st_size|st_block0|st_block1
1|a|0|0|973385730|973385730|973385730|0|0|0|0|0
24|f|500|500|973695537|973695537|973695537|973695537|40700|0|0|308|0
25|f|500|500|954365144|973695521|973695537|973695537|100600|0|28587|309|310
26|f|500|500|954365144|973695521|973695537|973695537|100600|0|340|338|0
2049|f|500|500|973695537|973695537|973695537|973695537|40700|0|0|8489|0
2050|f|500|500|953943572|973695536|973695537|973695537|100600|0|4178|8490|8491
2051|f|500|500|960098764|973695521|973695537|973695537|100600|0|52345|8495|8496
2052|f|500|500|953943572|973695537|973695537|973695537|100600|0|4860|8548|8549
2053|f|500|500|959130680|973695521|973695537|973695537|100600|0|28961|8553|8554
2054|f|500|500|959130680|973695521|973695537|973695537|100600|0|87647|8583|8584
2055|f|500|500|961959437|973695521|973695537|973695537|100600|0|30799|8670|8671
2056|f|500|500|959130680|973695521|973695537|973695537|100600|0|50176|8702|8703
2057|f|500|500|953943572|973695537|973695537|973695537|100600|0|21700|8752|8753
2058|f|500|500|959130680|973695521|973695537|973695537|100600|0|22865|8775|8776
2059|f|500|500|959130680|973695521|973695537|973695537|100600|0|14584|8799|8800
2060|f|500|500|953943572|973695521|973695537|973695537|100600|0|12276|8815|8816
2061|f|500|500|959130680|973695521|973695537|973695537|100600|0|10840|8827|8828
2062|f|500|500|959130680|973695521|973695537|973695537|100600|0|26027|8838|8839
```

Ko pridemo do številke *inode*, lahko pridemo do podatkov na katere kaže *inode* z ukazom **icat**.

```
examiner1% icat -f linux-ext2 ext2-bitstream.dd 2054
/*
dcc.c — handles:
activity on a dcc socket
disconnect on a dcc socket
...and that's it! (but it's a LOT)
dprintf'ized, 27oct95
*/
/*
This file is part of the eggdrop source code
copyright (c) 1997 Robey Pointer
and is distributed according to the GNU general public license.
```

```
For full details, read the top of 'main.c' or the file called
COPYING that was distributed with this code.
*/
#if HAVE_CONFIG_H
#include <config.h>
```

Tu lahko naletimo na probleme, ker Linux pri brisanju podatkov odstranjuje referenco z *inode* na sektorje na katerih so zapisani podatki. Obstaja še en pogosto uporabljen način pridobivanja izbrisanih podatkov s preverjanjem direktorijev in iskanjem izbrisanih vnosov, če le ti obstajajo. Obstajajo številna forenzična orodja za pridobivanje izbrisanih podatkov, katera uporabljajo en ali oba izmed opisanih načinov (Sleuth Kit, The Linux disk editor, debugfs, The SMART tool, The Autopsy Forensic Browser…). Še en neomenjen način pridobivanja zbrisanih podatkov je z uporabo *file carvinga*. *File carving* je postopek pridobivanja izbrisanih podatkov, ki uporablja razredne karakteristike. Ukaza **scalpel** in **foremost** sta pri tem najbolj pogosto uporabljena. Pri tem se sprehajamo čez *bitstream* kopijo diska in pregledujemo glave in noge datotek ter iščemo razredne karakteristike. Za primer, če se glava datoteke začne z D0 CF, potem smo našli skrito (izbrisano) .doc datoteko, ki se nadaljuje vse do EOF znaka.

Primer *file carvinga*:

```
examiner1% foremost –o carved-foremost –v floppycopy.dd
foremost version 0.62
Written by Kris Kendall and Jesse Kornblum.
Using output directory: /e1/carved-foremost
Verbose mode on
Using configuration file: foremost.conf
Opening /e1/linuxpractical.dd.
Total file size is 1474560 bytes
/e1/case2/floppycopy.dd: 100.0% done (1.4 MB read)
A doc was found at: 17408
Wrote file /e1/case2/carved-foremost/00000000.doc -- Success
A doc was found at: 37888
Wrote file /e1/case2/carved-foremost/00000001.doc -- Success
A jpg was found at: 76800
Wrote file /e1/case2/carved-foremost/00000002.jpg -- Success
A jpg was found at: 77230
Wrote file /e1/case2/carved-foremost/00000003.jpg -- Success
A jpg was found at: 543232
Wrote file /e1/case2/carved-foremost/00000004.jpg -- Success
A gif was found at: 990208
Wrote file /e1/case2/carved-foremost/00000005.gif -- Success
A jpg was found at: 1308160
Wrote file /e1/case2/carved-foremost/00000006.jpg -- Success
Foremost is done.
```

Linux operacijski sistem ima eno zelo koristno lastnost, da uporablja grupe blokov za shranjevanje podatkov s čimer se ustvarjajo gruče (*clusters*) na disku. V kolikor sumimo da je zlikovec zbrisal eno dnevniško datoteko z diska, tako lahko preberemo celoten blok z diska na katerega je shranjena vsebina */var/log* in poiščemo vse izbrisane datoteke.

Skrivanje podatkov

V nadaljevanju bo prikazanih nekaj načinov skrivanja podatkov - dva preprosta z manipulacijo imen datotek, skrivanje v *file slack*, steganografijo in z alternativnim tokom podatkov.

Najpreprostejši način skrivanja je preimenovanje končnice datoteke, npr. datoteko *child.gif* preimenujemo v *child.doc* . Ta način je najlažje odkriti, ker lahko pogledamo zgolj glavo datoteke, saj praviloma velja, da ima vsak tip datoteke svojo glavo. V tabeli so prikazani začetki glav za 3 priljubljene formate:

Table 15.4 Headers and Footers of Common File Types[a]

File Type	Header	Footer
JPEG	Usually FF D8 FF E0 or FF D8 FF E1 and sometimes FF D8 FF E3	FF D9
GIF	47 49 46 38 37 61 or 47 49 46 38 39 61	00 3B
Microsoft Office	D0 CF 11 E0 A1 B1 1A E1	N/A

[a]*Additional common file signatures are tabulated at http://www.garykessler.net/library/file_sigs.html.*

Če pogledamo glavo *child.txt* in vidimo, da se začne z 47 49 46, potem datoteka ni tekstovna, ampak slikovna z končnico *.gif*. Drugi način je prav tako enostaven. V kolikor koncu imena datoteke dodamo ~ (tilda), Linux označi datoteko kot *backup* in jo zaradi tega skrije. Podobno velja, če na začetek imena dodamo . (piko), ker Linux potem prepozna datoteko kot skrito in jo ne prikaže. Oba načina se da odkriti, če uporabljamo izpis s skriti datotekami.

Tretji način skrivanja je skrivanje podatkov v *file slack*. *File slack* je prostor med koncem datoteke in koncem sektorja. Pri sektorju velikosti 1024 bajtov in datoteki velikosti 1010 bajtov, imamo *file slack* velikosti 14 bajtov v katerega lahko skrijemo podatke. Podatki se lahko skrijejo z ukazom *bmap* in zastavico *slack*. Za primer bomo v *file slack* skrili informacijo *»cybercriminal«:*

```
[root@tortilla slack]# bmap --slack file1.txt
getting from block 139522
file size was: 10
slack size: 1014
block size: 1024
cybercriminal
```

Skrivanje podatkov v *file slack* se redko uporablja zaradi dveh razlogov:

- v *file slack* lahko skrijemo podatke majhne velikosti
- če spremenimo velikost datoteke, katera se nahaja v sektorju (dodamo znake), lahko povozimo skrite podatke v *file slacku*.

Četrti način skrivanja podatkov je steganografija. Steganografija je znanost skrivanja podatkov v informacijo na način, da do podatkov lahko prideta le pošiljatelj in prejemnik informacije.

Steganografija obstaja od antične zgodovine in se je razvijala v različnih oblikah (pisma, nevidna črnila...). Za digitalno forenziko je zanimava digitalna steganografija.
Digitalna steganografija uporablja skrivanje podatkov v obstoječe datoteke. Obstajajo 3 metode:

- Metoda vstavljanja – sporočilo se vstavi v datoteko na način da ne vpliva na uporabo datoteke, npr. za EOF znakom.
- Metoda generiranja – generiramo novo datoteko glede na sporočilo katerega skrivamo.
- Metoda zamenjave – sporočilo vnesemo v datoteko tako, da prepišemo najmanj pomembne bite posameznih bajtov.

Metoda zamenjave se posebej uporablja pri digitalnih slikah. Razlog je precej enostaven – če se spremenijo najmanj pomembne informacije (biti) slike, človeško oko ne more zaznati sprememb.
Obstajajo številna ogrodja za steganografijo, katera so sestavljena iz:

- podatkov katere želimo skriti,
- nosilca – datoteka v katero želimo skriti podatke kot je npr. digitalna slika v .png formatu (nosilec moramo izbrati na način tako, da so spremenjeni nosilec s skriti podatki in originalni nosilec dokaj podobni),
- verige – podatki so lahko skriti v nosilcu na različnih mestih,
- enkripcijski in dekripcijski algoritem.

Najbolj poznano prosto dostopno orodje za steganografijo je *OpenPuff*, ki omogoča skrivanje podatkov v slike, avdio in video zapise. Za skrivanje uporablja več enkripcijskih algoritmov – *CSPRNG*, *hash enkripcijo* in drugo. Za primer steganografije sta priložene dve sliki – v orginalno sliko na levi je bila podtaknjena slika na desni (maček).


Peti način skrivanja podatkov je uporaba alternativnega toka podatkov (*alternative data stream* – *ADS*). *ADS* je funkcija NTFS datotečnega sistema, ki je bila razvita, zaradi kompatibilnosti z *Machintosh* računalniki. Obstajajo virusi in trojanci, kateri izkoriščajo *ADS* za dodajanje ene datoteke v drugo datoteko, na način, da ni vidna uporabnikom. Obstajajo številna forenzična orodja, ki znajo zaznati podatke v *ADS*.

Najbolj siguren način skrivanja podatkov je uporaba močnih enkripcijskih algoritmov. Tudi če vemo da obstaja datoteka in kje se nahaja, kljub temu ne moremo priti do vsebine datoteke, če ne poznamo ključa za dekriptiranje.

Časovni žigi (timestamps) v Linux sistemih

Pri forenzični preiskavi diska so nam časovni žigi zelo pomembni, ker iz njih lahko dobimo veliko informacij o podatkih zapisanih na disku. V spodnji tabeli so opisani načini spremembe časovnih žigov pri manipulaciji podatkov na disku:

Action	Last Modified Date-Time	Last Accessed Date-Time	Inode Change Date-Time
File moved within a volume	Unchanged	Unchanged	Updated
File copied (destination file)	Updated	Updated	Updated

V Linux sistemih obstajajo 2 vrsti časov: *ctime* in *mtime*. *Ctime* je časovni žig, kateri se spreminja, če se delajo akcije z datoteko (npr. kopiranje datoteke), *mtime* je časovni žig, kateri se spreminja, če se spreminja vsebina datoteke. Enkrat, ko se datoteka izbriše, se prekine povezava med datoteko in njenim *inodom*. Ker *inode* ni dostopen za pregled v datotečnem sistemu, nam podatek, kdaj je zbrisana datoteka, lahko ostane zapisan v *inodu*, dokler se *inode* znova ne uporabi. Ko se datoteka doda ali odstrani iz mape, *inode* spremeni časovni žig direktorija in čase zadnjega dostopa in izmenjave datoteke. Zaradi tega se spremeni *ctime* direktorija, če se izbriše kakšna datoteka znotraj direktorija. To je lahko zelo koristno pri preiskavi, saj lahko primerjamo *ctime* direktorija in vseh njemu pripadajočih datotek, ter vidimo ali je znotraj direktorija izbrisana kakšna datoteka.

Dnevniške datoteke v Linux

V Linux operacijskih sistemih obstaja veliko število dnevniških datotek, katere beležijo podatke, ki nam olajšajo delo preiskovanja diska. Dnevniške datoteke se nahajajo v direktoriju */var/log*. Primeri dnevniških datotek:

/var/log/message: generalna sporočila in sporočila povezana z sistemom
/var/log/auth.log: avtentikacija
/var/log/kern.log: jedro
/var/log/cron.log: Cron
/var/log/maillog: poštni strežnik
/var/log/qmail/ : Qmail
/var/log/httpd/: Apache strežnik in sporočila o napakami
/var/log/boot.log : *System boot* sporočila
/var/log/mysqld.log: *MySQL server* sporočila
/var/log/secure: avtentikacija
/var/log/utmp ali */var/log/wtmp*: *login* sporočila

Dnevniške datoteke imajo velikokrat tudi aplikacije in se ponavadi nahajajo na mestu, kjer je shranjena aplikacija. Dnevniške datoteke so bogat vir informacij, ampak moramo z njimi delati pazljivo, ker se informacije v dnevniških datotekah lahko ponaredijo.

Disk zaščiten z gesli in enkripcijo

Podatki na disku so lahko zaščiteni z gesli ali z enkripcijo. V Linuxu obstaja ukaz **crpyt**, ki dela enostavno enkripcijo.

```
% crypt -key 'guessme' < plaintext> ciphertext
```

Klasične distribucije Linuxa so za enkripcijo uporabljale *DES* enkripcijski algoritem, ki se danes obravnava kot šibek enkripcijski algoritem, ker ga lahko enostavno dekriptiramo s pomočjo *brute-force* napada. Novejše distribucije Linuxa uporabljajo močne enkripcijske algoritme, kot *MD5* (po najnovejših raziskavah se tudi *MD5* smatra kot šibek), *Blowfish*, *SHA-256* ali *SHA-512*.

Pogosto se originalni tekst samo zbriše z diska, in z uporabo že opisanih metod lahko pridemo do besedila tudi brez dekriptiranja. Za dekripcijo lahko tudi poskusimo uganiti skriti ključ, saj se za ključ pogosto uporabi beseda, ki je zelo enostavna. Če lastnik diska nima veliko tehničnega znanja, potem nekje sigurno obstaja zapisan ključ. Pogosto se za skrite ključe in gesla uporabljajo besede, ki imajo za lastnika nekakšen pomen. Številni hekerji danes uporabljajo metode socialnega inženiringa na lastnikih diskov, da bi prišli do gesel.

Po nekaterih raziskavah, je še vedno več kot 40% vseh gesel na svetu, lahko uganljivih. Primeri takih gesel: *password*, *12345678*, *qwerty* in drugo.

Po zakoniku Republike Slovenije je lastnik diska dolžan preiskovalcem dati vsa gesla in skrite ključe, v nasprotnem primeru se ga lahko kaznuje.

Na internetu obstaja veliko število orodji za razbijanje enkripcije, najbolj poznana sta *Crack* in *Jack the Ripper*.

Primer raziskave diska

Forenzično preiskavo diska bomo prikazali na realnem primeru. Ko smo priklopili disk, lahko izpišemo vse direktorije na disku z ukazom **ls**

```
# ls -lat /mnt
total 73
drwxr-x---  17 root     root        1024 May  1 09:01 root
drwxrwxrwt   6 root     root        1024 May  1 04:03 tmp
drwxr-xr-x   8 root     root       34816 Apr 30 04:02 dev
drwxr-xr-x  34 root     root        3072 Apr 29 14:17 etc
drwxr-xr-x   2 root     root        2048 Apr 26 16:52 bin
drwxr-xr-x   2 root     root        1024 Apr 26 11:12 boot
drwxr-xr-x   3 root     root        3072 Apr 21 04:01 sbin
drwxr-xr-x   4 root     root        3072 Apr 21 03:56 lib
drwxrwxr-x   2 root     root        1024 Mar  3 13:27 cdrom
drwxr-xr-x   2 root     root        1024 Oct  9  1999 home
drwxr-xr-x   2 root     root       12288 Oct  9  1999 lost+found
drwxr-xr-x   4 root     root        1024 Oct  9  1998 mnt
drwxr-xr-x   2 root     root        1024 Oct  9  1999 proc
drwxr-xr-x  20 root     root        1024 Aug  2  1998 usr
```

```
drwxr-xr-x  18 root     root        1024 Aug  2  1998 var
```

Če preverimo imena direktorijev – *root, tmp, dev, etc, bin*..., lahko pridemo do zaključka, da je disk katerega preiskujemo zares korenski datotečni sistem. V */etc/fstab* so vse particije našega datotečnega sistema.

```
# less /mnt/etc/fstab
 . . .
/dev/hda1            /dosc            msdos   defaults      0 0
/dev/hda2            /                ext2    defaults      1 1
/dev/hda4            /home            ext2    defaults      1 2
/dev/hda3            swap             swap    defaults      0 0
/dev/cdrom           /cdrom           iso9660 noauto,user,ro 0 0
/dev/fd0             /floppy          ext2    noauto,user,rw 0 0
none                 /proc            proc    defaults      0 0
none                 /dev/pts         devpts  mode=0622     0 0
```

Lotimo se preiskave diska. Najprej preverimo, kaj je zapisano v */etc/passwd* v katerem so zapisani uporabniški računi. Med drugim tu lahko najdemo tudi račune, katere je ustvaril zlikovec.

```
# less /mnt/etc/passwd
 . . .
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
z:x:0:0::/:/bin/bash
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:
news:x:9:13:news:/var/spool/news:
uucp:x:10:14:uucp:/var/spool/uucp:
operator:x:11:0:operator:/root:
r00t:x:598:500:::/bin/bash
games:x:12:100:games:/usr/games:
y:x:900:100::/tmp:/bin/bash
gopher:x:13:30:gopher:/usr/lib/gopher-data:
ftp:x:14:50:FTP User:/home/ftp:
nobody:x:99:99:Nobody:/:
gdm:x:42:42::/home/gdm:/bin/bash
xfs:x:100:233:X Font Server:/etc/X11/fs:/bin/false
user1:x:500:500:User 1:/home/user1:/bin/tcsh
user2:x:501:501:User 2:/home/user2:/bin/tcsh
user3:x:502:502:User 3:/home/user3:/bin/tcsh
named:x:25:25:Named:/var/named:/bin/false
```

Preverimo, kateri računi nam izgledajo zelo sumljivi, kot *r00t* (sumljivo ime), *y* (domači imenik mu je */tmp*), *named* (zadnji račun) in x (uid=0 in gid=0, isto kot korenski uporabnik!).

Vzemimo za primer *y* za katerega preverimo kaj je v */tmp* direktoriju.

```
# ls -lat /mnt/tmp
total 156
drwxrwxrwt   6 root     root        1024 May  1 04:03 .
-r--r--r--   1 root     gdm           11 Apr 29 14:17 .X0-lock
drwxrwxrwt   2 root     gdm         1024 Apr 29 14:17 .X11-unix
drwxrwxrwt   2 xfs      xfs         1024 Apr 29 14:17 .font-unix
drwxr-xr-x  25 y        root        1024 Apr 28 23:47 ..
drwx------   2 user1    user1       1024 Apr 26 17:36 kfm-cache-500
-rw-rw-r--   1 user1    user1      12288 Apr 26 16:37 psdevtab
drwxrwxrwt   2 root     root        1024 Apr 21 11:12 .ICE-unix
-rwx------   1 root     root      138520 Apr 20 20:15 .fileMFpmnk
```

Zadnja datoteka ima zelo čudno ime in je izvršljiva. Z ukazom **string** preverimo, kaj je zapisano v datoteki.

```
# strings - /mnt/tmp/.fileMFpmnk
/lib/ld-linux.so.2
__gmon_start__
libpam.so.0
_DYNAMIC
_GLOBAL_OFFSET_TABLE_
pam_set_item
free
__ctype_tolower
malloc
strcmp
pam_end
pam_start
 . . .
File
Compressed
Block
Stream
[nowhere yet]
ftpd
:aAvdlLiop:P:qQr:sSt:T:u:wWX
bad value for -u
option -%c requires an argument
unknown option -%c ignored
 . . .
VirtualFTP Connect to: %s [%s]
banner
logfile
email
/var/log/xferlog
connection refused (server shut down) from %s
%s FTP server shut down -- please try again later.
lslong
/bin/ls -la
lsshort
lsplain
/bin/ls
```

```
greeting
full
terse
brief
%s FTP server (%s) ready.
%s FTP server ready.
FTP server ready.
. . .
FTP LOGIN REFUSED (already logged in as %s) FROM %s, %s
Already logged in.
/etc/ftphosts
FTP LOGIN REFUSED (name in %s) FROM %s, %s
anonymous
FTP LOGIN REFUSED (anonymous ftp denied on default server) FROM %s, %s
FTP LOGIN REFUSED (ftp in denied-uid) FROM %s, %s
/etc/ftpusers
```

Datoteka izgleda kot FTP strežnik, ki se ponavadi imenujejo ftpd ali .ftpd. Zaradi tega datoteko označimo kot sumljivo.

Potem je dobro preveriti nastavitvene datoteke v */dev* direktoriju, kjer se lahko nahajajo datoteke, katere je zlikovec pustil za sabo.

```
# cd /mnt/dev
# ls -lat | head -30
total 116
drwxr-xr-x   8 root     root       34816 Apr 30 04:02 .
srw-rw-rw-   1 root     root           0 Apr 30 04:02 log
crw-------   1 root     root      4,   1 Apr 29 14:17 tty1
crw-------   1 root     root      4,   2 Apr 29 14:17 tty2
crw-------   1 root     root      4,   3 Apr 29 14:17 tty3
crw-------   1 root     root      4,   4 Apr 29 14:17 tty4
crw-------   1 root     root      4,   5 Apr 29 14:17 tty5
crw-------   1 root     root      4,   6 Apr 29 14:17 tty6
srwxrwxrwx   1 root     root           0 Apr 29 14:17 gpmctl
srw-------   1 root     root           0 Apr 29 14:17 printer
crw-r--r--   1 root     root      1,   9 Apr 29 14:17 urandom
prw-------   1 root     root           0 Apr 29 14:14 initctl
drwxr-xr-x  25 y        root        1024 Apr 28 23:47 ..
crw-rw-rw-   1 root     tty       3,   2 Apr 28 11:44 ttyp2
crw-rw-rw-   1 root     tty       3,   0 Apr 28 11:43 ttyp0
crw-rw-rw-   1 root     tty       3,   1 Apr 28 11:43 ttyp1
-rw-r--r--   1 root     root          18 Apr 27 22:58 ptyp
drwxr-xr-x   4 r00t     root        1024 Apr 27 22:58 ...
crw-rw-rw-   1 root     tty       3,   4 Apr 27 12:02 ttyp4
crw-rw-rw-   1 root     tty       3,   3 Apr 27 11:56 ttyp3
crw-------   1 root     root      5,   1 Apr 21 11:09 console
lrwxrwxrwx   1 root     root           5 Apr 21 04:02 mouse -> psaux
drwxr-xr-x   2 root     root        1024 Apr 20 15:21 rev0
-rw-r--r--   1 root     root          33 Apr 20 15:21 ptyr
lrwxrwxrwx   1 root     root           9 Feb 28 02:23 isdnctrl -> isdnctrl0
lrwxrwxrwx   1 root     root           5 Feb 28 02:23 nftape -> nrft0
lrwxrwxrwx   1 root     root           3 Feb 28 02:23 fb -> fb0
```

```
lrwxrwxrwx   1 root     root          15 Feb 28 02:23 fd -> ../proc/self/fd
lrwxrwxrwx   1 root     root           4 Feb 28 02:23 ftape -> rft0
Broken pipe
```

Sumljive so nam datoteke *ptyp* in *ptyr* (- kot prvi znak v opisu datoteke), direktorij *rev0* in skriti direktorij ..

Preverimo kaj se nahaja v *ptyp*:

```
# less ptyr
 . . .
sp.pl
slice
ssynk4
rev0
bc1
snif
```

Datoteke, ki se nahajajo v *ptyp* so že poznane kot konfiguracijske datoteke za trojanske konje. Lahko preverimo, kje vse se uporabljajo datoteke:

```
# cd /mnt
# find . -ls | grep -f etc/ptyr
282058    1 drwxr-xr-x   2 root     root        1024 Apr 20 15:21 ./dev/rev0
282059    1 -rw-r--r--   1 root     root           5 Apr 20 15:21 ./dev/rev0/sniff.pid
282061   20 -rw-r--r--   1 root     root       19654 Apr 20 20:23 ./dev/rev0/tcp.log
164753    9 -rwxr-xr-x   1 1080     users       9106 Sep 20  1999 ./dev/rev0/slice
164754    8 -rwxr-xr-x   1 1080     users       8174 Sep 20  1999 ./dev/rev0/smurf4
164755    8 -rwxr-xr-x   1 1080     users       7229 Sep 20  1999 ./dev/rev0/snif
164756    4 -rwxr-xr-x   1 1080     users       4060 Mar  5  1999 ./dev/rev0/sp.pl
164770    9 -rwxr-xr-x   1 root     1000        8268 Aug 10  1999 ./dev/.../blitznet/slice2
 61907    2 -rwxr-xr-x   1 root     root        2006 Mar 29  1999 ./usr/bin/sliceprint
255230    1 -rw-r--r--   1 root     root         900 Mar 21  1999 ./usr/include/python1.5/sliceobject.h
```

Nekatere od njih so sigurno legitimne sistemske datoteke, a ker se uporabljajo v */dev*, gremo en korak naprej in preverimo */dev*.

```
# cd /mnt/dev
# less ptyp
 . . .
3 egg
3 egg
3 bnc
```

Obstaja trojanski konj, kateri skriva procese *egg* in *bnc* v izpisu ukaza **ps**. Preverimo kje se nahajajo izvršne datoteke z temi nazivi.

```
# cd /mnt/dev
# ls -lR ...
...:
total 2699
```

```
drwxr-sr-x   2 root     1000        1024 Aug 10  1999 blitznet
-rw-r--r--   1 root     root       30720 Apr 26 04:07 blitznet.tar
-rwxrw-r--   1 r00t     user1      22360 Apr 27 22:58 bnc
-rw-r--r--   1 900      users    2693120 Apr 20 22:18 collision.tar
-rw-rw-r--   1 r00t     user1        976 Apr 27 22:58 example.conf
-rw-rw-r--   1 user1    user1          5 Apr 28 20:35 pid.bnc

.../blitznet:
total 22
-rw-r--r--   1 root     1000        3450 Aug 10  1999 README
-rw-r--r--   1 root     1000        1333 Aug 10  1999 blitz.c
-rw-r--r--   1 root     1000        3643 Aug 10  1999 blitzd.c
-rwxr-xr-x   1 root     1000        2258 Aug 10  1999 rush.tcl
-rwxr-xr-x   1 root     1000        8268 Aug 10  1999 slice2
```

Direktorij */dev/rev0* se nahaja v izpisu. Lahko ga preverimo.

```
# ls -lR rev0
rev0:
total 51
-rwxr-xr-x   1 1080     users       9106 Sep 20  1999 slice
-rwxr-xr-x   1 1080     users       8174 Sep 20  1999 smurf4
-rwxr-xr-x   1 1080     users       7229 Sep 20  1999 snif
-rw-r--r--   1 root     root           5 Apr 20 15:21 sniff.pid
-rwxr-xr-x   1 1080     users       4060 Mar  5  1999 sp.pl
-rw-r--r--   1 root     root       19654 Apr 20 20:23 tcp.log
```

```
# cd /mnt/usr/bin
# ls -lat | head
total 89379
drwxr-xr-x   6 root     root       27648 Apr 21 04:01 .
-rwsr-xr-x   1 root     root       20164 Apr 15 19:23 chx
lrwxrwxrwx   1 root     root           8 Feb 28 02:28 netscape-navigator -> netscape
drwxrwxr-x   2 news     news        1024 Feb 28 02:25 rnews.libexec
drwxrwxr-x   2 news     news        1024 Feb 28 02:25 control
drwxrwxr-x   2 news     news        1024 Feb 28 02:25 filter
lrwxrwxrwx   1 root     root           4 Dec 30 13:06 elatex -> etex
lrwxrwxrwx   1 root     root           5 Dec 30 13:06 lambda -> omega
lrwxrwxrwx   1 root     root           3 Dec 30 13:06 latex -> tex
Broken pipe
```

```
# strings - chx
/lib/ld-linux.so.2
__gmon_start__
libcrypt.so.1
libpam.so.0
 . . .
/var/log/btmp
/usr/share/locale
util-linux
fh:p
```

login: -h for super-user only.
usage: login [-fp] [username]
/dev/tty
%s??
/dev/vcs
/dev/vcsa
login
login: PAM Failure, aborting: %s
Couldn't initialize PAM: %s
FAILED LOGIN %d FROM %s FOR %s, %s
Login incorrect
TOO MANY LOGIN TRIES (%d) FROM %s FOR %s, %s
FAILED LOGIN SESSION FROM %s FOR %s, %s
Login incorrect
.hushlogin
%s/%s
/var/run/utmp
/var/log/wtmp
/bin/sh
TERM
dumb
HOME
/usr/local/bin:/bin:/usr/bin
PATH
/sbin:/bin:/usr/sbin:/usr/bin
SHELL
/var/spool/mail
MAIL
LOGNAME
DIALUP AT %s BY %s
ROOT LOGIN ON %s FROM %s
ROOT LOGIN ON %s
LOGIN ON %s BY %s FROM %s
LOGIN ON %s BY %s
You have %smail.
new
login: failure forking: %s
setuid() failed
No directory %s!
Logging in with home = "/".
login: no memory for shell script.
exec
login: couldn't exec shell script: %s.
login: no shell: %s.
%s login:
login name much too long.
NAME too long
login names may not start with '-'.
too many bare linefeeds.
EXCESSIVE linefeeds
Login timed out after %d seconds
/etc/securetty
/etc/motd
/var/log/lastlog
Last login: %.*s

```
from %.*s
on %.*s
LOGIN FAILURE FROM %s, %s
LOGIN FAILURE ON %s, %s
%d LOGIN FAILURES FROM %s, %s
%d LOGIN FAILURES ON %s, %s
 . . .
```

Sporočila o napakah in referenca na *hushlogin* nam pove, da gre za trojanskega konja (verzija *login)*. Informacije o simbolih so vključene v prevedene objekte, če niso odstranjene. Preverimo:

```
# nm chx
chx: no symbols
```

Preverimo, kaj se nahaja v dinamičnih knjižnicah.

```
# ldd chx
        libcrypt.so.1 => /lib/libcrypt.so.1 (0x40018000)
        libpam.so.0 => /lib/libpam.so.0 (0x40045000)
        libdl.so.2 => /lib/libdl.so.2 (0x4004d000)
        libpam_misc.so.0 => /lib/libpam_misc.so.0 (0x40050000)
        libc.so.6 => /lib/libc.so.6 (0x40054000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Zdaj vidimo, da se je uporabljal *crypt()* in *Pluggable Authentication Module (PAM)*, kar nedvomno nakazuje na trojanskega konja.
Na koncu raziskave še enkrat izračunamo *MD5 hash* kodo in primerjamo z originalno kodo, da zagotovimo nespremenljivost podatkov. S tem smo zaključili preiskavo diska.

Forenzična analiza omrežja

Do nedavnega je veljalo, da je bilo pri preiskavi potrebno preverjati zgolj računalnik in predvsem disk posameznika. V zadnjem času, ko pa je omrežje postalo svetovno razširjeno in velika večina ljudi uporablja e-maile, socialna omrežja, internetne trgovine, pretakanje preko spleta in druge omrežne storitve je pregledovanje postalo precej bolj prepleteno tudi z omrežjem. Forenzični preiskovalci se morajo tako naučiti še enega področja in poznati dodatna orodja. Med drugim morajo poznati standarde za internetne protokole, brskalnike, e-maile, prenos datotek kar jim je še dodatno oteženo zaradi hitro razvijajočega področja. Možno je celo, da preiskovalci sploh nimajo dostopa do posameznega računalnika s čimer celotna analiza poteka preko spleta oz. spletnih storitev in s pomočjo internetnih ponudnikov, kreditnih kartic, telefonskih pogovorov ter drugih.

Postopki preiskovanja trdega diska za digitalnimi dokazi so dobro definirani. Ko pa se srečamo z omrežjem pa lahko nastopijo nepričakovane ovire. Podatkov na omrežju je ogromno in so dinamični, kar nam onemogoča zajem celotnega stanja v določenem trenutku.
V nasprotju z preiskovanjem enega računalnika preiskovalci ne morejo preprosto izklopiti celotnega interneta, saj morajo kljub preiskavi zagotavljati, čimbolj nemoteno delovanje omrežja. Poleg tega to tudi ne bi bilo smiselno, saj bi se ob izklopu izgubili vsi podatki. Naslednja težava je, da je na omrežju praktično nemogoče izolirati kraj zločina, saj kriminalec lahko napada iz večih mest hkrati.

Porazdelitev na večjih delih omrežja pa je lahko tudi pozitivno, saj je tako težko uničiti vse dokaze. Podjetja velikokrat delajo varnostne kopije, ki jih hranijo na različnih lokacijah, kar pomeni, da po vsej verjetnosti ne bodo uničili vseh podatkov.

UNIX sistemi so večinoma konfigurirani tako, da beležijo in hranijo uporabniške podatke za datoteke, e-maile in gesla za oddaljene dostope. Zato je pametno iskati sledi za oddaljene dostope na omrežju, ki lahko vodijo do dodatnih virov dokazov. Prvotnega pomena je, da se to naredi kar se da hitro še predno se sledi izgubijo.

UNIX sistemi v */etc/hosts* pogosto hranijo spisek povezav, ki pogosto komunicirajo med sabo. Za UNIX okolje so značilni lokalni in omrežni skupni diski, kateri so hranjeni v */etc/fstab*, saj so avtomatično pripeti ob zagonu. Podobna informacija se nahaja tudi v */etc/mtab* in */proc/mounts*, ki pa hranita tudi informacije o pripetih napravah preostalih posameznih uporabnikov. Poleg NFS se lahko s pomočjo Samba orodij dostopa tudi do drugih omrežnih virov na Windows okolju. UNIX računalnike lahko konfiguriramo tako, da sistemske podatke pošiljamo na oddaljen sistem s spremembo v datoteki */etc/rsyslog.conf*, */etc/syslog.conf*. To naredimo z dodajanjem vrstice **.* @IP:PORT*. Podobno velja za tiskanje, kjer se hranijo informacije v datotetki */etc/printcap*.

Pri tem postopku je pomembno, da na oddaljene lokacije dostopamo fizično po standardnih postopkih in ne brez pravne podlage.

Digitalni dokazi na TCP/IP plasteh

TCP/IP je skupni jezik vseh omrežij, osnova interneta in je poznan kot de facto standard. Je enostavnejši od OSI modela in med sabo združi nekatere plasti. Razdeljen je na 4 plasti:

- Aplikacijska plast med drugim vsebuje protokole telnet, ftp, http, smnp, smtp.
- Transportni plasti sta najpomembnejša protokola TCP in UDP.
- Mrežna plast ima protokole kot so internetni protokol (IP), ICMP in drugi.

- Fizični in povezavni plasti pa so glavni ARPANET, paketni radio in LAN.

Naloga fizične plasti je, da skrbi za fizični prenos podatkov. Mrežna plast skrbi za transparentno pošiljanje podatkov med mrežami. Dostava pri tem ni zagotovljena, niti vrstni red dostave. Povezava s povezavnim slojem je protokol ARP. Prenosna plast skrbi za povezavni in brezpovezavni način delovanja. TCP predstavlja tok podatkov med procesom na različnih računalnikih. Aplikacijska plast služi uporabi standardnih (npr. pošta, splet, IRC in drugo) in nestandardnih aplikacij. Te aplikacije uporabljajo storitve spodnjih plasti.

Dokaze sicer lahko dobimo iz več naprav kot so računalniki, usmerjevalniki in drugih.

Fizična/povezavna plast

Fizična in povezavna plast nudi temelj za vse kar najdemo na omrežju.

Najbolj pogost način za zbiranje dokazov na omrežju je t.i. prisluškovanje omrežju. Ta način zbiranja dokazov je primerljiv s tistim, ko naredimo kopijo trdega diska.

- Z orodjem **ifconfig** lahko skonfiguriramo omrežne naprave. Uporablja se predsem ob zagonu za nastavitev vmesnikov ter kasneje za preverjanje ali nastavljanje. Z ukazom **ifconfig -a** lahko preverimo stanje vseh naprav. S tem tudi vidimo IP in MAC naslove.
- Orodje **arp** se uporablja za nastavljanje oz. prikaz ARP tabele relacij med IP in MAC naslovi. Omogoča tudi brisanje oz. dodajanje zapisov. ARP oz. Address Resolution Protocol sicer služi preslikavi naslovov iz omrežne v povezavno plast.

Naslove računalnikov lahko dobimo tudi iz ARP tabel ali zgodovine DHCP na usmerjevalniku. DHCP med drugim hrani informacije o MAC naslovu, IP naslovu, času dodelitve in odvzema ter imenu računalnika.

Omrežna in transportna plast

Vsak komunikacijski sistem potrebuje mehanizem naslavljanja. Pogosto, ni pa nujno je potreben tudi nek sistem preverjanja, da je sporočilo prišlo na cilj. Omrežna in transportna plast sta skupaj odgovorni za pravilno dostavljanje paketov. Veliko količino shranjenih informacij se pridobi prav iz teh dveh plasti, kar nam zelo služi v digitalnem preiskovanju. Pomembni na tej plasti so predvsem IP naslovi in z njimi povezana DNS imena, ki nam pomagajo določiti izvor zločina.

- Orodje **nslookup** nam omogoča pretvorbo med IP naslovom in DNS imenom, kar lahko vidimo na Slika 5.

```
george@ubuntu:~/faks/RF$ nslookup www.fri.uni-lj.si
Server:         192.168.145.2
Address:        192.168.145.2#53

Non-authoritative answer:
Name:   www.fri.uni-lj.si
Address: 212.235.188.25

george@ubuntu:~/faks/RF$ nslookup 212.235.188.25
Server:         192.168.145.2
Address:        192.168.145.2#53

Non-authoritative answer:
25.188.235.212.in-addr.arpa     name = www.fri.uni-lj.si.

Authoritative answers can be found from:
188.235.212.in-addr.arpa        nameserver = ns.fri.uni-lj.si.
188.235.212.in-addr.arpa        nameserver = ns.uni-lj.si.
188.235.212.in-addr.arpa        nameserver = metulj.uni-lj.si.
ns.uni-lj.si    internet address = 193.2.64.45
metulj.uni-lj.si        internet address = 193.2.64.46
```

Slika 5: Prikaz uporabe orodja nslookup

- Z **dig** lahko poizvedujemo o pripadajočih DNS strežnikih. Občasno pri tem lahko dobimo tudi informacije o drugih računalnikih, ki uporabljajo DNS strežnik, a strežniki tega večinoma ne dopuščajo, saj napadalci lahko to s pridom izkoriščajo.
- Druge še zelo uporabno orodje je tudi **traceroute**, ki nam omogoča sledenje paketov po vozliščih, ki jih obišče paket, da doseže ciljni IP. To nam v digitalnem preiskovanju lahko pride zelo prav, saj večinoma poti ostajajo iste, kar nam omogoča, da naredimo dodatno preiskovanje na vozliščih ali DNS strežnikih, kjer se prav tako beležijo podatki o prenosih.
- Aplikacije za svojo delovanje uporabljajo določena vrata. Za preverjanje ali pa določena aplikacija deluje lahko uporabimo orodje **nmap**. To orodje nam dodatno omogoča tudi npr. odkrivanje operacijskega sistema ali verzij programov, ki so lahko povezani tudi z določenimi varnostnimi luknjami. V začetku nam lahko zelo dobro služi za odkrivanje karakterističnih značilnosti v omrežju.
- Orodje **netstat** nam lahko izpiše aktivne povezave našega računalnika, kar nam ob pomoči orodja **nmap** lahko služi kot določanje aktivnih aplikacij kot je npr. VNC ali skupni vir z drugim računalnikom. Na ta način lahko pridemo do dodatnih virov informacij, ki jih je potrebno raziskati.

Aplikacijska plast

- Pri preiskovanju na omrežjih je lahko velika količina podatkov nerelevantnih za določen primer. V tem primeru je pametno dobiti zgolj podatke, ki nas zanimajo, kjer si pomagamo npr. z filtriranjem glede na čas, IP naslov, neuspešne poskuse prijave ali glede na obravnavan primer primerne kriterije. Primer lahko vidimo na Slika 6, ki uporablja nativno orodje Linuxa **tcpdump** oz. Slika 7, ki uporablja orodje **tshark**.
- Za oddaljen dostop je najstarejše orodje **telnet**, ki deluje preko TCP protokola. V osnovi nudi dostop do ukazne vrstice operacijskega sistema, kjer komunikacija poteka tekstovno.
- Za prenos datotek se uporablja orodje **ftp** s katerim lahko iz lokalnega računalnika naložimo datoteke na ali iz oddaljenega sistema. S stališča računalniške forenzike nam to lahko doprinese dodatni vir podatkov za pregledovanje.

```
tcpdump > tcpdump_log & ping www.google.com & (sleep 5; pkill tcpdump; pkill ping)
grep -a ICMP tcpdump_log
```

Slika 6: Prikaz filtriranja iz izhoda ukaza tcpdump

```
tshark -i eth0 > tshark_log & ping www.google.com & (sleep 5; pkill tshark; pkill ping)
grep ICMP tshark_log
```

Slika 7: Prikaz filtriranja iz izhoda ukaza tshark

Preiskovanje spletnih brskalnikov v operacijskem sistemu Linux

Preiskovanje spletnega brskalnika Firefox

Spletni brskalnik Firefox shranjuje podatke o uporabnikovem brskanju v datoteke tipa *sqlite*. SQLite je datoteka narejena v programskem jeziku C, ki vsebuje relacijske podatkovne baze. Brskalnik Firefox shranjuje sqlite datoteke z informacijami o uporabniškem brskanju v mapi:

~/.mozilla/firefox/<PROFILE>/

Med drugimi imamo v mapi naslednje datoteke:

- addons.sqlite – podatki o nameščenih dodatkih,
- cookies.sqlite – podatki o shranjenih piškotih,
- downloads.sqlite – podatki o snetih datotekah, ki se še nahajajo v oknu Prenosi,
- extensions.sqlite – podatki o razširitvah,
- formhistory.sqlite – podatki o uporab. vnešenih podatkih v vnosna polja v obrazcih,
- places.sqlite – podatki o zgodovini brskanja,
- search.sqlite – podatki o iskalnih nizih,
- ter ostale.

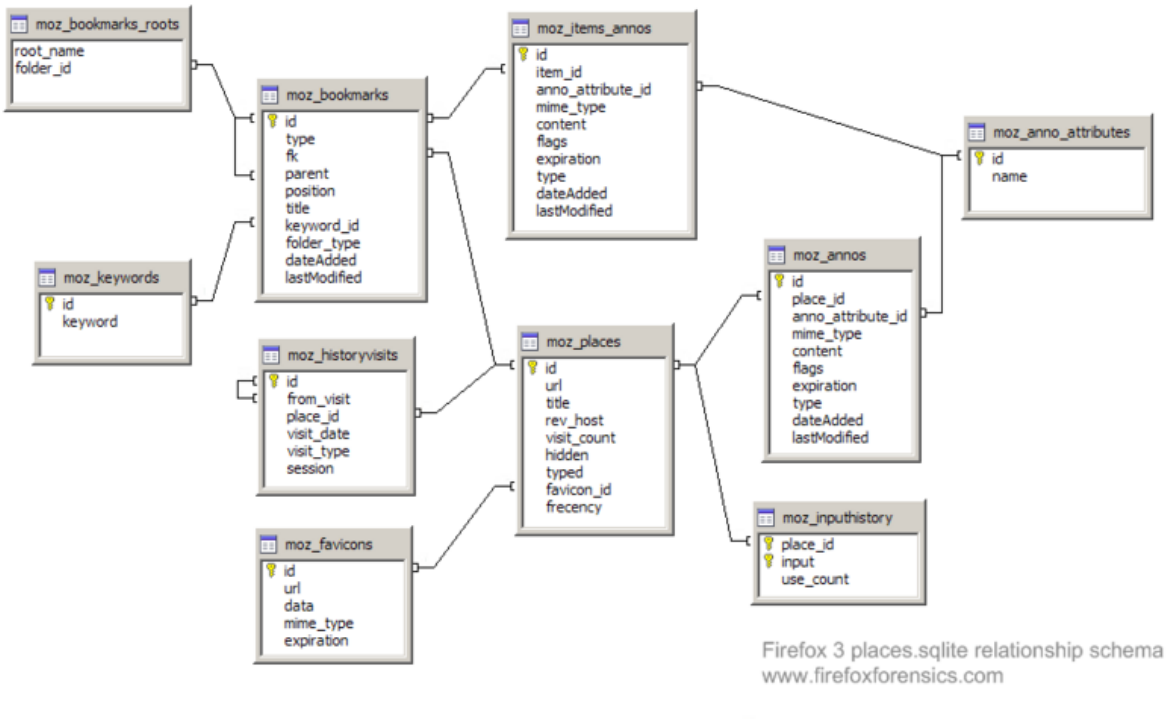
Datoteka *Cookies.sqlite* vsebuje tabelo *moz_cookies*, ki pa je sestavljena naslednji shemi.

moz_cookies [id; name; value; host; path; expiry; lastAccessed; isSecure; isHttpOnly; baseDomain; creationTime]

Datoteka *Formhistory.sqlite* vsebuje tabelo *moz_formhistory.*

moz_formhistory [id; fieldname; value; timesUsed; firstUsed; lastUsed; guid]

Relacijska shema podatkovne baze znotraj datoteke *places.sqlite* je bolj komplicirana in je prikazana na sliki 1, v njej pa so shranjeni podrobni podatki o zgodovini brskanja uporabnika.



Slika : Relacijska shema znotraj places.sqlite

Časovni žigi znotraj podatkovnih tabel so tipa *PRTime*. Gre za število mikrosekund od 1.1.1970 dalje, zapisanih v 64-bitnem številskem tipu. Stolpec *visit_type* označuje tip prehoda na spletno stran (ali je uporabnik vpisal naslov spletne strani v url okence, ali je bil preusmerjen preko povezave ipd.).

Vsebino *sqlite* datotek je možno prebirati s pomočjo orodja SQLite. S pomočjo spodnjega ukaza, lahko uporabnik na svoj operacijski sistem namesti orodje sqlite3, ki je le eden od orodij za pregledovanje sqlite datotek.

apt-get install sqlite3

Uporabnik lahko, za preiskovanje sqlite datotek, namesti in uporablja tudi orodja z grafičnim vmesnikom. Nekatera znana orodja so *SQLiteMan* ali *SQLiteBrowser*. Prav tako je mogoče namestiti *SQLite Manager*, dodatek za spletni brskalnik Firefox. Po namestitvi lahko orodje odpremo preko brskalnika Firefox (Tools -> SQLite Manager).

Preiskovalec za preiskovanje sqlite datoteke lahko uporablja *sqlite3* ali katerega od ostalih SQLite brskalnikov. Za analizo tabel najprej poženemo orodje sqlite3 na način, da v terminalu vpišemo sqlite3 <ime_datoteke.sqlite>.

Po zagonu sqlite preidemo v program. S pomočjo ukaza

sqlite>.tables

vidimo vse podatkovne tabele shranjene znotraj datoteke. Z ukazom

sqlite>.schema <ime_tabele>

vidimo shemo podatkovne tabele. Ostale možnosti uporabe sqlite orodja vidite s pomočjo ukaza .help.

Za preiskovanje tabel uporabljamo *strukturirani povpraševalni jezik SQL*.

Spodnji primer prikazuje pridobivanje zgodovine brskanja skupaj z url naslovom, naslovom strani in časom, pretvorjenim v ljudem bolj berljiv zapis.

```
sqlite>SELECT datetime(moz_historyvisits.visit_date/1000000,
'unixepoch') as date, moz_places.url
 ...>FROM moz_places, moz_historyvisits
 ...>WHERE moz_places.id = moz_historyvisits.place_id
 ...>ORDER BY date DESC;
```

Spodnji primer prikazuje vse obiskane strani v nekem časovnem okviru:

```
sqlite>SELECT datetime((visit_date/1000000), 'unixepoch',
'localtime') as date, p.url, p.title
 ...>FROM moz_places p, moz_historyvisits
 ...>WHERE date BETWEEN '2012-04-02 00:00:00' AND '2012-04-04 00:00:00'
 ...>ORDER BY date DESC;
```

Preiskovanje spletnega brskalnika Google Chrome

Na operacijske sistemu Linux lahko uporabnik uporablja *Google Chrome* ali *chromium-browser*. Google Chrome temelji na odprtokodnem projektu Chromium. Brskalnika se v manjši meri ločita med sabo, pomembno je vedeti, da uporabljata različne mape za shranjevanje uporabniškega profila.

Spletni brskalnik Chromium shranjuje svojo zgodovino brskanja v mapo chromium znotraj:

~/.config/chromium/

Medtem, ko Google Chrome za shranjevanje uporablja mapo google-chrome:

~/.config/google-chrome/

Zgodovino in ostale informacije prav tako shranjujeta v sqlite datoteko (kljub temu, da datoteke nimajo .sqlite končnice), ki jo lahko analiziramo na podoben način, kot pri brskalniku Firefox.

Znotraj mape google-chrome se nahajajo med drugimi tudi naslednje datoteke:

- Favicons – informacije o ikonah spletnih strani,
- History – informacije o zgodovini brskanja,
- Bookmarks – informacije o zaznamkih,
- Archived History – informacije o arhivirani zgodovini,
- Top Sites,
- Cookies – informacije o piškotkih,
- in še mnogo ostalih. Uporabnik lahko pregleda mapo z ukazom *ls -al*.

Struktura tabel znotraj datoteke *History* je sledeča:

- downloads – vsebuje seznam pobranih datotek.
- presentation
- urls – vsebuje podatke o obiskanih spletnih straneh.
- keyword_search_terms
- segment_usage
- visits – vsebuje informacije o obiskih (časovni žigi, ...)
- meta – vsebuje metapodatke
- segments

Uporabnik za preiskovanje sqlite datoteke lahko uporablja sqlite3 ali katerega od ostalih SQLite brskalnikov. Za analizo tabel najprej poženemo orodje sqlite3 na način, da v terminalu vpišemo sqlite3 <ime_datoteke>. V kolikor je ime datoteke ločeno s presledkom, moramo ime datoteke navesti v narekovaje, primer: sqlite3 "Top Sites".

Pomembno je omeniti, da časovni žigi znotraj tabel niso povsod v epoch zapisu (primer tabela *visits*). V takšnih primerih so časovni žigi zapisani v mikrosekundah od 1.1.1601 dalje. Nekatere tabele imajo časovne žige v PRTime zapisu, število sekund od 1.1.1970 (primer downloads).

Znotraj tabele *visits* se nahaj stolpec transition, ki opisuje način, kako je uporabnik prišel na stran, podobno kot je to opisano pri brskalniku Firefox. Obstaja 11 t.i. prehodnih tipov med stranmi. 0 – pomeni, da je uporabnik prišel na stran preko povezave, 1 – uporabnik je vpisal url povezavo v brskalnik, potem sledijo še začetna stran, oddaja obrazcev, osvežitev strani itd.

Spodnji primer prikazuje pridobivanje podatkov o zgodovini brskanja (url naslov in naslov strani), prav tako opravimo pretvorbo časovnega žiga v ljudem bolj berljiv format.

```
sqlite>SELECT datetime(((visits.visit_time/1000000)-11644473600), "unixepoch") as time, urls.url, urls.title
 ...>FROM urls, visits
 ...>WHERE urls.id = visits.url
 ...>ORDER BY time;
```

Časovnemu žigu odštejemo število sekund od 1.1.1601 do 1.1.1970.
S pomočjo LIMIT <št.prikazanih vrstic> lahko omejimo rezultat na zadnjih nekaj zapisov iz tabele.

Omenimo lahko še program *log2timeline*, ki iz datotek zgodovine brskanja zgradi sosledje dogodkov za lažje razumevanje uporabnikovega brskanja.

Povzetek brskalnikov

Kot vidimo mora preiskovalec poleg poznavanja v kakšni obliki brskalnik shranjuje svojo zgodovino in kje jo shranjuje, poznati tudi jezik SQL, s katerim si lahko pomaga pri pridobivanju podatkov iz datotek. Seveda so za lažjo uporabo na spletu tudi grafična orodja za manipulacijo sqlite tabel, vendar bo vsaj splošno znanje SQL jezika preiskovalcu veliko v pomoč.

Pomembno je tudi omeniti, da imajo brskalniki možnost vklopa prikritega/privatnega (InPrivate) načina brskanja s katerim naj ne bi beležili zgodovine brskanja in ostalih podatkov v sqlite datoteke. Gre za način brskanja, ki ne shranjuje zgodovine oz. informacij v datoteke, če pa že, pa jih ob izhodu iz zasebnega načina ali zaprtju brskalnika, izbriše. Tak primer so piškotki, ki se izbrišejo ob zaprtju

brskalnika. Vendar pa gre bolj za lokalno prikritost informacij, kajti podatki, ki se zapisujejo v registre in t.i. 'cache data', so nedotaknjeni, preko katerih forenziki še zmeraj lahko odkrijejo indice.

Zaključek

V tem članku smo opisali različne aspekte digitalne forenzike operacijskega sistema Linux. Preiskovali smo diske, omrežja in brskalnike (Chrome, Firefox) in pri tem skušali čimbolj opisati korake preiskave. Pri tem smo poskušali uporabiti čim več nativnih orodij, ki jih ponuja Linux operacijski sistem. Za vsako od teh opisanih področij smo podali različne primere uporabe in ob tem pridobili veliko novih znanj iz vseh področij.

Viri in reference:

1. Eoghan Casey. (2011). Digital Evidence and Computer Crime: Forensic Science, Computers and the Internet. Third Edition. Maryland, USA: Elsevier
2. Mozilla Firefox 3 History File Format. (maj 2012). Povezava: http://www.forensicswiki.org/wiki/Mozilla_Firefox_3_History_File_Format
3. Google Chrome Forensics. (maj 2012). Povezava: http://computer-forensics.sans.org/blog/2010/01/21/google-chrome-forensics/
4. Log2timeline. (maj 2012). Povezava: http://log2timeline.net/
5. Robertson, G., »An Introduction to Hiding and Finding Data on Linux« - GIAC Security Essentials Certification, SANS Institute, 2003.
6. Artz, D.,«Digital steganography: hiding data within data", Internet Computing, volume 5 issue 3, 75-80, 2001.
7. Huebner, E., Bem, D., Kai Wee, C., "Data hiding in the NTFS file system", Digital investigation, volume 3 issue 4, 211-226, 2006.
8. Dittrich, D., "Basic steps in forensics analysis of Unix systems", http://staff.washington.edu/dittrich/misc/forensics/, datum pristopa: 13.5.2012.