HRVATSKA AKADEMSKA I ISTRAŽIVAČKA MREŽA
CROATIAN ACADEMIC AND RESEARCH NETWORK

Primjena TCT programskog paketa u postupcima forenzičke analize

CCERT-PUBDOC-2004-01-56

CARNet CERT u suradnji s LS&S

Sigurnosni problemi u računalnim programima i operativnim sustavima područje je na kojem CARNet CERT kontinuirano radi.
Rezultat toga rada ovaj je dokument koji je nastao suradnjom CARNet CERT-a i LS&S-a, a za koji se nadamo se da će Vam koristiti u poboljšanju sigurnosti Vašeg sustava.

CARNet CERT, www.cert.hr - nacionalno središte za **sigurnost računalnih mreža** i sustava.

LS&S, **www.lss.hr**- laboratorij za sustave i signale pri Zavodu za elektroničke sustave i obradbu informacija Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu.

Sadržaj

1. Uvod

Temeljito prikupljanje i detaljna analiza podataka sa kompromitiranih sustava svakako je jedan od osnovnih uvjeta za uspješno rješavanje incidenta i vjernu rekonstrukciju događaja na sustavu. Osim o tipu i značenju, prilikom prikupljanja podataka posebnu je pažnju potrebno voditi o njihovoj postojanosti, budući da neki podaci imaju kraći vijek trajanja od drugih. Na primjer, podatke u radnoj memoriji sustava potrebno je prikupiti ranije od onih na tvrdom disku, budući da postoji veća vjerojatnost od njihovog gubitka. Također je potrebno voditi računa da sam postupak prikupljanja i analize podatka ne utječe na njihov integritet, pogotovo ukoliko se isti podaci žele iskoristiti kao dokazni materijal. Budući da ručno prikupljanje i analiza podataka predstavlja prilično mukotrpan proces, s vremenom su razvijeni specijalizirani alati koji sigurnosnim stručnjacima olakšavaju ovaj postupak. Jedan od takvih alata je upravo TCT programski paket opisan u ovome dokumentu.

TCT programski paket sastoji se od niza alata od kojih je svaki namijenjen određenoj funkciji. Jezgru programa čini `grave-robber`, program pisan u Perl programskom jeziku, iz kojeg se pokreću svi ostali alati, no moguće je i njihovo zasebno korištenje. Iako program omogućuje jednostavnije i automatizirano prikupljanje podataka, potrebna je određena razina znanja i iskustva kako bi se mogućnosti programa iskoristile u potpunosti.

Osim analize pojedinih programa koji dolaze u sklopu TCT programskog paketa s primjerima njihovog korištenja, na samom početku ukratko su opisane osnovne karakteristike datotečnog sustava kod Unix/Linux operacijskih sustava, budući da njihovo poznavanje igra važnu ulogu u postupcima rekonstrukcije i prikupljanja podataka. U tom smislu opisana je osnovna struktura datotečnog sustava, smisao i značenje *inode* struktura te `mactime` vremenske oznake.

2. Linux/Unix datotečni sustav

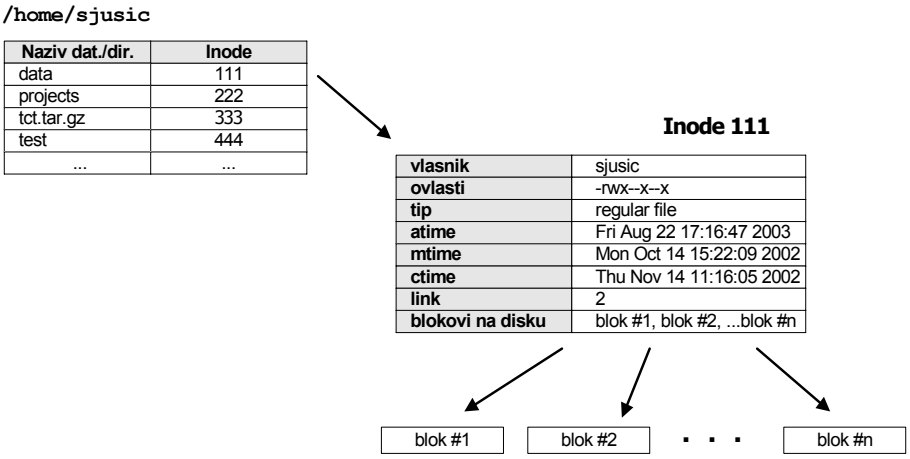
Budući da je TCT programski paket namijenjen isključivo forenzičkoj analizi računala s Unix/Linux operacijskim sustavima, na samom početku potrebno je dati nekoliko osnovnih napomena vezanih uz karakteristike i strukturu datotečnog sustava kod spomenutih operacijskih sustava. Poznavanje strukture datotečnog sustava vrlo je važan segment u postupku provođenja forenzičke analize, budući da većina koraka vezanih uz ovaj postupak ovisi upravo o analizi i prikupljanju podataka sa tvrdog diska kompromitiranog računala (iako postoje i brojni drugi izvori podataka koje je potrebno uzeti u obzir, npr. radna memorija sustava i sl.). Iako je danas dostupan velik broj alata koji omogućuju automatizirano prikupljanje podataka sa kompromitiranih sustava (jedan od takvih alata je upravo i The Coroner's Toolkit alat), poznavanje osnovnih principa i interne strukture datotečnog sustava gotovo je neophodno za ispravnu upotrebu ovih alata i maksimalno iskorištavanje njihovih mogućnosti.

Općenito gledajući, bez obzira o kojem se operacijskom sustavu radi, uloga datotečnog sustava je ista. Ideja je da se podaci koji su na tvdom disku pohranjeni u obliku niza jedinica i nula korisniku prezentiraju na smislen i logičan način koji će omogućiti njihovo korištenje. Datotečni sustav se najbolje može opisati kao funkcionalnost operacijskog sustava koja se ponaša kao sučelje između korisnika i tvrdog diska, putem kojeg se omogućuje pristup podacima. Osim organizacije podataka, datotečni sustav posjeduje i druge funkcionalnosti, kao što su ovlasti pristupa, vlasništvo nad datotekama i direktorijima, bilježenje vremena pristupa i modifikacije podataka i sl. Kako će kasnije biti pokazano, neka od navedenih svojstava su ključna prilikom provođenja forenzičke analize.

Slično kao i kod drugih operacijskih sustava, Unixov datotečni sustav organiziran je u obliku stabla (engl. *tree hierarchy*), na vrhu kojeg se nalazi korijenski root direktorij. Unutar root direktorija dalje se nalaze ostali poddirektoriji i datoteke, što u konačnici čini ranije spomenutu strukturu. Treba napomenuti da datoteke kod Linux operacijskog sustava mogu biti različitog tipa (simbolička veza, *socket*, FIFO datoteka, direktorij, regularna datoteka i dr.).

Datoteka (engl. *file*), kao jedan od osnovnih elemenata datotečnog sustava kod Unix operacijskih sustava, ima poseban značaj. Razlog tomu je taj što se gotovo svi podaci o sustavu i pripadajućim programima korisniku prezentiraju upravo putem datoteka (stanje pocesa, mrežne konekcije, uređaji, konfiguracijske datoteke, log zapisi i sl.). Svakoj takvoj datoteci su putem datotečnog sustava pridjeljeni određeni parametri (vlasnik, ovlasti pristupa, vremena pristupa i dr.), kojima se opisuju njena svojstva i prava pristupa. Sve ove informacije pohranjenje su u strukturi poznatoj pod nazivom *inode*, koja čini osnovni element Unix/Linux datotečnog sustava. Svaka datoteka posjeduje svoj vlastiti *inode* zapis (osim u slučaju veza – *links*, što je također jedno od karakterističnih svojstava Unix operacijskih sustava), u kojem su pohranjeni osnovni podaci o datoteci, zajedno s pokazivačima na fizičke blokove na disku u kojima je pohranjen njen sadržaj. Upravo se zbog ovakvog pristupa vrlo često kaže da Unix datoteke nemaju imena, već *inode* oznake. *Inode* zapisi sadrže sve relevantne podatke o svakoj datoteci, osim samog sadržaja datoteke koji je razmješten po fizičkim blokovima na disku (na koje naravno pokazuje svaka *inode* struktura).

Direktorij, kao sljedeći važniji element datotečnog sustava, može se jednostavno opisati kao kontejner za datoteke i poddirektorije koji se u njemu nalaze, i sastoji se od jednostavne liste imena datoteka, odnosno direktorija i pripadajućeg rednog broja *inode* strukture u kojoj se nalaze podaci o tom zapisu. Preciznije rečeno, direktorij je specijalna datoteka koja sadrži određeni broj parova **ime datoteke(direktorija)<-> inode broj**. Opisani koncept prikazan je na sljedećoj slici (Slika 1). Broj i oznaka pojedinih *inode* struktura, zajedno s veličinom i brojem fizičkih blokova na disku određuje se prilikom formatiranja tvrdog diska i pohranjuje se u tzv. *superblock* zapis koji je smješten u drugom bloku particije na kojoj je kreiran datotečni sustav. *Superblock* zapis sadrži općenite informacije o datotečnom sustavu kao što su broj *inode* struktura i fizičkih blokova na disku, stanje pojedinih *inode* struktura i sl.



Slika 1: Osnovni koncept Unix/Linux datotečnog sustava

Razumijevanje opisanog odnosa datoteka, direktorija, *inode* struktura i fizičkih blokova na disku vrlo je važno prilikom provođenja forenzičke analize, ili preciznije rečeno, obnavljanja jednom izbrisanih datoteka. Naime, brisanje datoteke u ovom slučaju ne podrazumijeva niti brisanje blokova na disku, niti brisanje *inode* struktura, već uklanjanje zapisa koji povezuje ime datoteke sa pripadajućom *inode* strukturom. Iako nakon brisanja zapis sam po sebi više ne postoji, *inode* struktura i sam sadržaj datoteke još se uvijek nalazi negdje na tvrdom disku (sve dok operacijski sustav ponovo ne alocira oslobođenu *inode* strukturu i blokove na disku). To znači da je do sadržaja izbrisane datoteke moguće doći identificiranjem oslobođene *inode* strukture, ali i do svih drugih podataka koji su zapisani unutar *inode* zapisa (vlasnik, vremena pristupa i sl.).

Broj *inode* strukture u kojoj su pohranjene informacija o odgovarajućoj datoteci moguće je dobiti zadavanjem ls naredbe, sa prosljeđenim parametrom `-i`:

```
# ls  -i
162104 README
161988 apache
162105 cron
162065 halt
162125 inetd
162113 klogd
162184 nfs-kernel-server
162193 postfix
162127 ppp
162067 reboot
161991 samba
162080 sendsigs
162133 setserial
162186 ssh
162130 sysklogd
```

Sistemski pozivi `stat`, `lstat`, `fstat` omogućuju pristup podacima pohranjem u *inode* strukturi. Također postoji i naredba `stat` koja koristi navedene sistemske pozive i omogućuje jednostavniji dolazak do ovih podataka. U nastavku je dan primjer korištenja stata naredbe.

```
# stat /etc/fstab
  File: "/etc/fstab"
  Size: 299             Blocks: 8          IO Block: 4096   Regular File
Device: 301h/769d       Inode: 210636      Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: Tue Feb 17 15:10:23 2004
Modify: Tue Nov  4 15:29:33 2003
Change: Tue Nov  4 15:29:33 2003
```

Iz dobivenog ispisa jasno se mogu vidjeti podaci o navedenoj datoteci (vlasnik, ovlasti pristupa, *inode*, vremena pristupa i modifikacije, broj *inode* strukture i sl.).

Korištenjem iste naredbe moguće je doći i do općenitih informacija o datotečnom sustavu pohranjenih u ranije spomenutom *superblock* zapisu.

```
# stat -l /dev/hda7
  File: "/dev/hda7"
  Size: 0               Blocks: 0          IO Block: 4096   Block Device
Device: 301h/769d       Inode: 324140      Links: 1     Device type: 3,7
Access: (0660/brw-rw----)  Uid: (    0/    root)   Gid: (    6/    disk)
Access: Tue Nov  4 15:27:09 2003
Modify: Thu Mar 14 22:51:02 2002
Change: Tue Nov  4 15:27:09 2003
```

3. MAC vremenske oznake

Kako je spomenuto u prethodnom poglavlju, *inode* struktura između ostalih zapisa sadrži i podatke o vremenima pristupa i modifikacije datoteke, odnosno direktorija. U tom smislu postoje tri vremena koje sustav bilježi i njihovo razumijevanje također je vrlo važno prilikom provođenja forenzičke analize. Radi se osljedećim vremenima:

- Vrijeme zadnje modifikacije sadržaja datoteke (`mtime`),
- Vrijeme zadnjeg pristupa datoteci (`atime`),
- Vrijeme zadnje modifikacije *inode* strukture (`ctime`),

Prema prvim slovima kratica navedenih vremenskih oznaka, ova vremena zajedno se nazivaju MAC vremena. Prilikom analize kompromitiranih sustava ova vremena igraju vrlo važnu ulogu, pogotovo kada je potrebno precizno rekonstruirati ranije događaje te odrediti njihov vremenski slijed. Iako iskusniji korisnici vrlo lako mogu lažirati ova vremena (npr. korištenjem naredbe `touch` ili mijenjanjem trenutnog vremena na sustavu), njihova detaljna analiza vrlo često daje konkretne i precizne podatke o proteklim događajima. `ctime` parametar ima poseban značaj, budući da označava promjene na samoj `inode` strukturi. Dok je vremena pristupa i zadnje modifikacije sadržaja moguće jednostavno lažirati, s `ctime` parametrom to je nešto teže, budući da se bilo koja modifikacija ili pokušaj lažiranja opet bilježi unutar `ctime` parametra.

Vrijednosti ovih parametara također je moguće vrlo jednostavno dobiti korištenjem `ls` naredbe (iste podatke moguće je dobiti korištenjem naredbe `find`). Naredba `ls` inicijalno prikazuje `mtime` vrijeme, prosljeđivanjem parametra `-u` prikazuje se vrijeme `atime`, dok prosljeđivanje opcije `-m` rezultira prikazom `ctime` vremena. Npr. sljedećom naredbom moguće je vidjeti ispis datoteka u trenutnom direktoriju s pripadajućim vremenima njihove zadnje modifikacije (inicijalno ponašanje ls naredbe):

```
#ls -al s*
-rwxr-xr-x    1 root     root         2215 Nov 22  2002 samba
-rwxr-xr-x    1 root     root          358 Feb 26  2001 sendsigs
-rwxr-xr-x    1 root     root         3797 Nov  7  2001 setserial
-rwxr-xr-x    1 root     root          499 Feb 26  2001 single
-rwxr-xr-x    1 root     root         1771 Apr  8  2002 skeleton
-rwxr-xr-x    1 root     root         1699 Jun 28  2002 ssh
-rwxr-xr-x    1 root     root         1908 Jan  3  2002 sysklogd
```

Zadavanjem naredbe `touch` promijenit će se vrijeme zadnje modifikacije datoteke pod nazivom `ssh` tako da izgleda kao da je sadržaj datoteke mijenjan zadnji puta 25. travnja 2003. u 18:00h. Za sintaksu korištenja nredbe `touch` preporučuje se pregledavanje pripadajuće man stranice.

```
# touch -m 0425180003 mountall.sh
```

Nakon ove promjene zadavanje `ls` naredbe daje sljedeći ispis, iz kojeg se može vidjeti da je pripadajuće vrijeme promijenjeno:

```
#ls -al s*
-rwxr-xr-x    1 root     root         2215 Nov 22  2002 samba
-rwxr-xr-x    1 root     root          358 Feb 26  2001 sendsigs
-rwxr-xr-x    1 root     root         3797 Nov  7  2001 setserial
-rwxr-xr-x    1 root     root          499 Feb 26  2001 single
-rwxr-xr-x    1 root     root         1771 Apr  8  2002 skeleton
-rwxr-xr-x    1 root     root         1699 Apr 25  2003 ssh
-rwxr-xr-x    1 root     root         1908 Jan  3  2002 sysklogd
```

No, zadavanjem naredbe `ls` sa parametrom `-c` može se uočiti kako je *inode* zapis iste datoteke zadnji puta promijenjen dana 17. veljače, što je datum pisanja ovog dokumenta. Ovaj podatak upućuje na potencijalno lažiranje podataka vezanih uz spomenutu `ssh` datoteku.

```
# ls -al s*
-rwxr-xr-x    1 root     root         2215 Nov 22  2002 samba
-rwxr-xr-x    1 root     root          358 Feb 26  2001 sendsigs
-rwxr-xr-x    1 root     root         3797 Nov  7  2001 setserial
-rwxr-xr-x    1 root     root          499 Feb 26  2001 single
-rwxr-xr-x    1 root     root         1771 Apr  8  2002 skeleton
-rwxr-xr-x    1 root     root         1699 Apr 25  2003 ssh
-rwxr-xr-x    1 root     root         1908 Jan  3  2002 sysklogd
```

4. The Coroner's Toolkit programski paket

TCT programski paket sastoji se od skupa alata, od kojih su neki pisani u Perl, a neki u C programskom jeziku. Osnovna im je namjena prikupljanje i analiza podataka sa kompromitiranih sustava. Program nije vezan uz točno određeni zadatak ili funkciju, tako da je teško u jednoj rečenici opisati sve njegove mogućnosti i kvalitete. Ugrađene funkcionalnosti korisnicima omogućuju automatizirano prikupljanje informacija sa kompromitiranih sustava te jednostavniju rekonstrukciju prethodnih događaja. Važno je napomenuti da je TCT alat namijenjen statičkoj analizi sustava, što znači da se analizira stanje računala u trenutku pokretanja programa. Sve aktivnosti koje se u tom trenutku odvijaju na sustavu (kopiranje datoteka, mrežne konekcije, stanja procesa) TCT program neće zabilježiti. Osnovne komponente koje čine program su:

- `Grave-robber`,
- `Mactime`,
- `Unrm`,
- `Lazarus`.

U nastavku dokumenta biti će zasebno opisane mogućnosti i način rada svakog od navedenih programa, zajedno s primjerima njihovog korištenja.

4.1. Grave-robber

`Grave-robber` program predstavlja jezgru TCT programskog paketa i njegova je primarna namjena **prikupljanje** informacija s kompromitiranog računala. Program je realiziran kao Perl skripta koja pokreće ostale podprograme zadužene za pojedine zadatke, od kojih je većina napravljena u obliku Perl modula smještenih u `lib` poddirektoriju TCT programskog paketa. Način rada programa, tip i količina podataka koji se žele prikupljati, zajedno s njihovom lokacijom, može se kontrolirati uređivanjem konfiguracijske datoteke programa (`$TCT_HOME/conf/grave-robber.conf` i `coroner.conf`) ili prosljeđivanjem odgovarajućih opcija programu.

Program je realiziran tako da prvo prikuplja one podatke za koje je poznato da imaju kraći "vijek trajanja" u odnosu na ostale podatke. Naime, dobro je poznato da na računalu postoje različite kategorije podataka s obzirom na njihovu postojanost (npr. radna memorija u odnosu na tvrdi disk), a `grave-robber` program ovu činjenicu uzima u obzir. Program prvo prikuplja informacije iz radne memorije kao što su podaci o pokrenutim procesima, stanjima mrežnih konekcija i sl., nakon čega slijedi postupak prikupljanja podataka s tvrdog diska. Inicijalni direktorij pregledavanja datotečnog sustava moguće je zadati navođenjem njegovog imena prilikom pokretanja programa. Bez eksplicitno navedenog imena direktorija, program prikuplja podatke u odnosu na root (/) direktorij, što je ujedno i preporučljiv način njegova korištenja. Iako je program moguće pokrenuti pod bilo kojim korisničkim računom, najbolje ga je pokretati pod ovlastima `root` korisnika, budući da većina aktivnosti programa zahtijeva najviše ovlasti na sustavu.

4.1.1. Pokretanje programa

Opcije `grave-robber` programa moguće je podijeliti u tri osnovne skupine. Svaka od ovih skupina, zajedno s nekim osnovnim parametrima, opisana je u nastavku:

- **General**

Opcije vezane uz općenito ponašanje programa. Slijedi kratki opis nekih od osnovnih opcija iz ove skupine:

`-b body_file` - lokacija gdje program pohranjuje rezultate izvršavanja md5 funkcije nad datotekama sustava, zajedno sa *inode* podacima dobivenim izvršavanjem `lstat()` sistemskog poziva (inicijalna vrijednost ovog parametra je `$TCT_HOME/data/hostname/body`).

`-c corpse_dir` – opcija koja se koristi za zadavanje inicijalnog direktorija kada se analiza ne provodi na "živom" sustavu, već na kopiji tvrdog diska ili particije (npr. korištenjem `mount` naredbe). Ukoliko se opciji `-c` kao argument proslijedi točka montiranja diska ili particije, program će pretpostaviti da se ispod navedene točke nalazi hijerarhija direktorija tipična za Linux/Unix operacijske sustave. Npr. `/etc` direktorij biti će tražen na mjestu `/mnt/disk/etc`.

`-D datadir` – direktorij u kojem program pohranjuje prikupljene informacije (inicijalna vrijednost `$DATA/hostname/`.

`-c errorfile` – navođenje datoteke u koju se bilježe greške generirane tijekom izvođenja programa.

`-v` – količina informacija koju program generira za vrijeme izvođenja (engl. *verbosity level*).

- **micro data collection opcije**

Opcije namijenjene preciznijem definiranju skupine podataka koje program prikuplja (MAC vremena, informacije o pokrenutim procesima, *inode* zapisi i sl.).

`-F` – prikupljanje podataka.

`-i` – prikupljanje *inode* podataka sa nealociranog prostora na disku.

`-I` – prikupljanje informacija o trenutno pokrenutim procesima na sustavu.

`-M` – izračunavanje md5 funkcije nad datotekama. Ovi podaci vrlo su korisni prilikom provođenja forenzičke analize, budući da omogućuju provjeru integriteta prikupljenih informacija.

`-m` – izvršavanje `lstat()` sistemskog poziva nad datotekama kao ulaznim podacima za mactime program.

`-O` – pohrana datoteka koje su još otvorene, a ujedno i izbrisane s tvrdog diska. Najčešće se radi o konfiguracijskim datotekama, izvršnim programima i sl.).

`-p` – Kopiranje memorije procesa u datoteku (putem `pcat` naredbe). Ovu opciju se preporučuje koristiti s velikim oprezom, budući da može uzrokovati probleme u radu sa sustavom.

`-P` – pokretanje naredbi vezanih uz pokrenute procese na sustavu (ps, lsof i sl.), u svrhu dolaska do informacija potrebnih za izradu kopije izvršnih programa. Opcija se koristi na sustavima gdje do kopije programa nije moguće doći na temelju informacija iz `/proc` datotečnog sustava.

`-s` – pokretanje općenitih naredbi na sustavu vezanih uz prikupljanje informacija o samom poslužitelju (`netstat`, `df`, `arp`, `ifconfig`, `uname`, `ksyms`, `ipcs`, `ps` i dr.).

`-S` – ovom opcijom pohranjuju se datoteke definirane `save_this_file` parametrom unutar konfiguracijske datoteke programa.

- **macro data collection opcije**

Posebna kategorija opcija koje grupiraju pojedine mikro opcije u logičke skupine. Opcije u ovoj kategoriji sadrže nekoliko kombinacija mikro opcija za koje se smatra da korisnicima najčešće mogu zatrebati. Na taj način korisniku se olakšava učestalo pokretanje programa sa istim opcijama.

`-E` - opcija kombinira inicijalne postavke programa zajedno s opcijama `-I` i `-p`.

`-f` – tzv. *fast capture* način rada. Bez analize datotečnog sustava i generiranja MD5 zapisa. Podrazumijeva opcije `-O`, `-P` i `-s`.

`-n` – podrazumijeva inicijalne opcije programa. Podrazumijeva opcije `-i`, `-m`, `-M`, `-P`, `-s`, `-t`, `-I`, `-O`, `-F`, `-S`, `-V`.

4.1.2. Prikupljeni podaci

Ovisno o zadanim opcijama, program će na definiranoj lokaciji napraviti strukturu datoteka i direktorija u kojima se nalaze podaci prikupljeni sa kompromitiranog sustava. Bez dodatnih opcija program sve prikupljene informacije pohranjuju u direktorij pod nazivom `data/hostname`, gdje ime `hostname` odgovara imenu stroja na kojem se provodi analiza (ovaj direktorij je u stvari samo simbolička veza na pravi direktorij sa podacima čije ime odgovara imenu sustava i vremenu kada je `grave-robber` program pokrenut.

U nastavku je opisana struktura `data/hostname` direktorija, zajedno s pripadajućim datotekama i direktorijima.

- **`command_out`**

Poddirektorij u kojem se nalaze rezultati izvršavanja naredbi sustava pokrenutih od strane `grave-robber` programskog paketa (`arp`, `df`, `ifconfig`, `netstat`, `lsmod`, `ipcs`, `top`, `uname`, `w`, `who`, `uptime` i sl.). Imena datoteka odgovaraju imenima izvršenih naredbi. Uz svaku datoteku također se generira i pripadajuća `<ime_naredbe>.md5` datoteka u kojoj se nalazi rezultat izvršavanja MD5 funkcije nad dobivenom datotekom.

- **`conf_vault`**

Poddirektorij u kojem su pohranjene kopije svih važnijih datoteka na sustavu. Datoteke koje će biti ovdje pohranjene moguće je definirati uređivanjem `save_this_files`, `coroner.cf` i

`grave-robber.cf` datoteka. U istom direktoriju nalazi se i datoteka pod nazivom `index.html`, koja omogućuje pristup svim pohranjenim datotekama putem Web sučelja(Slika 2).

Slika 2: HTML stranica sa vezama na identificirane datoteke sustava

- **`proc`**

Poddirektorij u kojem su pohranjene kopije (engl. *image*) svih procesa aktivnih na sustavu u trenutku pokretanja programa (informacije dobivene zadavanjem opcije `-I`). Za svaki proces kreira se zasebna datoteka čije ime sadrži PID procesa i vrijeme kada je program pokrenut. Budući da se radi o binarnim datotekama, nije ih moguće pregledavati tekstulanim editorima, već je potrebno koristiti specijalizirane programe predviđene za ovu namjenu (`objdump`, `objcopy`, `strings` i sl.). Npr. korištenjem `objdump` programa moguće je analizirati strukturu "snimljenog" procesa.

```
# objdump -headers 958.out._2004_02_19_12\:52\:57_+0100
958.out._2004_02_19_12:52:57_+0100:     file format elf32-i386

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .interp       00000013  080480f4  080480f4  000000f4  2**0
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  1 .note.ABI-tag 00000020  08048108  08048108  00000108  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  2 .hash         00000388  08048128  08048128  00000128  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  3 .dynsym       000007f0  080484b0  080484b0  000004b0  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .dynstr       00000446  08048ca0  08048ca0  00000ca0  2**0
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  5 .gnu.version  000000fe  080490e6  080490e6  000010e6  2**1
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  6 .gnu.version_r 00000040  080491e4  080491e4  000011e4  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  7 .rel.dyn      00000180  08049224  08049224  00001224  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  8 .rel.plt      00000268  080493a4  080493a4  000013a4  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  9 .init         00000018  0804960c  0804960c  0000160c  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
 10 .plt          000004e0  08049624  08049624  00001624  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
 11 .text         0000d3c4  08049b10  08049b10  00001b10  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
 12 .fini         0000001c  08056ed4  08056ed4  0000eed4  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
 13 .rodata       00002e40  08056f00  08056f00  0000ef00  2**5
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
 14 .data         00001064  0805a000  0805a000  00012000  2**5
                  CONTENTS, ALLOC, LOAD, DATA
 15 .eh_frame     00000004  0805b064  0805b064  00013064  2**2
                  CONTENTS, ALLOC, LOAD, DATA
```

```
 16 .dynamic      000000c8  0805b068  0805b068  00013068  2**2
                  CONTENTS, ALLOC, LOAD, DATA
 17 .ctors        00000008  0805b130  0805b130  00013130  2**2
                  CONTENTS, ALLOC, LOAD, DATA
 18 .dtors        00000008  0805b138  0805b138  00013138  2**2
                  CONTENTS, ALLOC, LOAD, DATA
 19 .jcr          00000004  0805b140  0805b140  00013140  2**2
                  CONTENTS, ALLOC, LOAD, DATA
 20 .got          00000204  0805b144  0805b144  00013144  2**2
                  CONTENTS, ALLOC, LOAD, DATA
 21 .bss          00002560  0805b360  0805b360  00013360  2**5
```

Nakon općenitih informacija o programu, biti će ispisani podaci o pojedinim dijelovima memorije zauzetim od strane dotičnog procesa. Npr. text sekcija označava područje memorije u kojem se nalazi izvršni kod programa, sekcije .sbss i .bss označava dio memorije u kojem su smještene neinicijalizirane varijable programa i sl.

- **`removed_but_running`**

U ovom direktoriju nalaze se kopije svih datoteka sa sustava koje su izbrisane, ali još uvijek otvorene. Imena datoteka sadrže ime uređaja na kojem se nalazi izbrisana datoteka, broj pripadajuće *inode* strukture te vrijeme u kojem je stvorena (npr. `_dev_hda7_10.out_2004_02_19_12:52:57_+0100`).

- **`pcat`**

Direktorij u kojem se nalaze kopije svih procesa dobivene `pcat` programom (Poglavlje 10). Ovaj direktorij biti će prisutan samo ukoliko je programu proslijeđena opcija `-p`. Budući da `pcat` program informacije o pojedinom procesu prikuplja iz radne memorije sustava, na ovaj način moguće je doći do vrlo korisnih informacija o pojedinom procesu (IP adrese, korisničke zaporke i sl.).

- **`user_vault`**

Direktorij u kojem se pohranjuju svi osjetljivi podaci vezani uz pojedine korisničke račune na sustavu (SSH datoteke, `history` i sl.).

- **`coroner_log`**

Datoteka u kojoj su zapisane sve naredbe izvršene tijekom rada programa. Uz svaku naredbu zabilježeni su i pripadajući argumenti te vrijeme izvođenja. `Coroner.log` datoteka nalazi se iznimno u korijenskom direktoriju TCT programa, a ne u `data/hostname` direktoriju, kao što je to bio slučaj s ranije opisanim datotekama.

- **`error.log`**

Datoteka sa svim porukama o greškama javljenim za vrijeme izvođenja programa. Slično kao i `coroner.log`, `error.log` datoteka nalazi se u korijenskom direktoriju TCT programa.

- **`body`**

Tzv. *mactime database* datoteka u kojoj su pohranjene informacije o svim pregledanim datotekama. Za svaku datoteku zabilježeni su njeni *inode* podaci (vremena pristupa, ovlasti pristupa, vlasnik i sl.). Primjer zapisa pohranjenog u `body` datoteci dan je u nastavku (jedan zapis inače odgovara jednom retku `body` datoteke, no zbog duljine zapisa, isti je ovdje prelomljen.

```
258a6ad4a23fce61d12b7cdc68bd4e19|/sbin/arp|769|605114|33261|-rwxr-xr-
x|1|0|0|0|45112|1077160237|1029924448|1047043677|4096|96
```

U sljedećoj tablici opisana su redom značenja pojedinih parametara unutar zapisa (parametri su međusobno odvojeni znakom `|`).

Ime parametra	Vrijednost	Značenje
md5	258a6ad4a23fce61d12b7cdc68bd4e19	Rezultat provođenja MD5 funkcije nad datotekom
file	/sbin/arp	Ime datoteke
st_dev	769	Uređaj na kojem se nalazi datoteka
st_ino	605114	Broj *inode* strukture
st_mode	33261	*Inode* protection oznaka
st_ls	-rwxr-xr-x	Ovlasti pristupa
st_nlink	1	Broj veza na datoteku
st_uid	0	UID oznaka vlasnika datoteke
st_gid	0	GID oznaka vlasnika datoteke
st_rdev	0	Tip uređaja (bitno za *inode* zapise specijalnih datoteka)
st_size	45112	Veličina datoteke u oktetima
st_atime	1077160237	Vrijeme zadnjeg pristupa

st_mtime	1029924448	Vrijeme zadnje modifikacije
st_ctime	1047043677	Vrijeme zadnje modifikacije *inode* zapisa
st_blksize	4096	Optimalna veličina bloka za zapis datoteke
st_blocks	96	Broj fizičkih blokova zauzetih za pohranu sadržaja datoteke (veličine 512 okteta).

Tablica 1: Opisa zapisa body datoteke

- **body.S**

Datoteka identična upravo opisanoj `body` datoteci, ali sa podacima o SUID datotekama identificiranim na sustavu.

- **MD5_all**

Datoteka sa md5 sumama svih datoteka generiranim od strane `grave-robber` programa.

- **MD5_all.md5**

Datoteka sa MD5 sumom upravo spomenute `MD5_all` datoteke.

Kao što se može vidjeti iz upravo opisanih karakteristika `grave-robber` programa, može se zaključiti kako isti omogućuje prilično jednostavno i automatizirano prikupljanje podataka sa kompromitiranog sustava. Uređivanjem konfiguracijskih datoteka i prosljeđivanjem odgovarajućih opcija moguće je utjecati na količinu i tip informacija koje se žele analizirati. Ovo je vrlo važno svojstvo, budući da specifičnosti analize mogu zahtijevati određeni tip podataka. Prikupljanje svih podataka, osim što bi u tom slučaju produljilo vrijeme trajanja postupka, također bi rezultiralo i znatno većim količinama podataka koje mogu otežati uočavanje bitnih informacija. Za potpuno iskorištavanje mogućnosti `grave-robber` programa, osim poznavanja njegovih karakteristika, potrebno je i određeno iskustvo u radu s njime.

5. Mactime

`Mactime` je vrlo jednostavan i moćan program koji omogućuje prikupljanje i analizu informacija o MAC vremenima pojedinih datoteka i direktorija (Poglavlje 3). Program se može pokretati ili zasebno ili u kombinaciji s `grave-robber` programom, odnosno sa `body` i `body.S` datotekama generiranim od strane `grave-robber` programa. `Mactime` program za dolazak do odgovarajućih vremena pristupa koristi `stat()` i `lstat()` pozive sustava. Osim općenitih parametara kojima je moguće preciznije definirati način rada programa, program dodatno prima i parametre `time1` i `time2` kojima se određuje vremenski period za koji se provodi analiza. U nastavku je dan primjer korištenja `mactime` programskog paketa (vremena time1 i time2 dana su u formatu `mjesec/dan/godina`):

```
#./mactime 01/09/04-01/10/2004
Jan 09 04 10:40:47     4096 m.c drwxr-xr-x root     root     /etc/rc.d/rc2.d
                       4096 m.c drwxr-xr-x root     root     /etc/rc.d/rc1.d
                       4096 m.c drwxr-xr-x root     root     /etc/sysconfig
```

Iz dobivenog ispisa može se vidjeti da su u danom periodu (preciznije dana 09. siječnja) modificirane svega tri datoteke. Uz svaku datoteku dan je znakovni niz koji opisuje koji je od vremenskih parametara promijenjen. Znak `m` označava `mtime` parametar, znak `a atime` parametar, a znak `c ctime` parametar.

Neke od općenitih opcija `mactime` programa navedene su u nastavku:

`-b` – alternativna lokacija `body` datoteke koja se koristi kao izvor podataka (bez ove opcije koristi se inicijalna lokacija definirana `coroner.cf` konfiguracijskom datotekom).

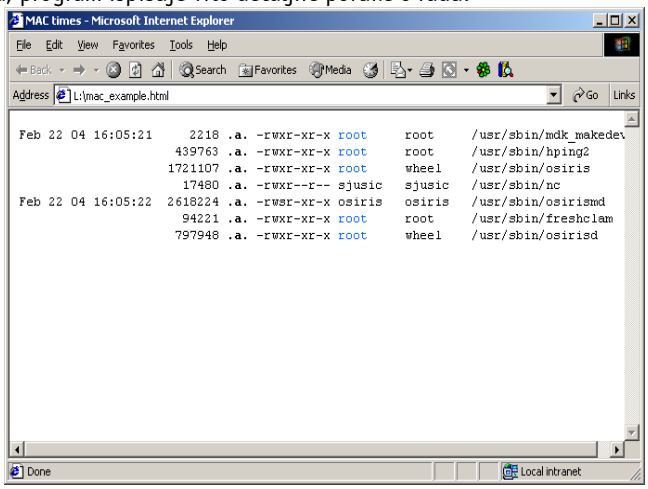
`-d` – prikupljane i analiza MAC vremena provodi se u odnosu na zadani direktorij, a ne na temelju `body` datoteke generirane od strane `grave-robber` programskog paketa. Ova opcija koristi se u slučajevima kada se `mactime` program koristi neovisno od `grave-robber` programa.

`-R` – rekurzivno pregledavanje u odnosu na direktorij zadan opcijom `-R`.

`-h` – ispis u HTML formatu (Slika 3).

`-u` – datoteke u vlasništvu navedenog korisnika obilježit će se drugom bojom (Slika 3).

`-D` – *debug* opcija; program ispisuje vrlo detaljne poruke o radu.



Slika 3: Ispis `mactime` programa u HTML formatu.

6. Unrm

Unrm program omogućuje rekonstrukciju podataka iz nealociranog područja datotečnog sustava. Podaci prikupljeni `unrm` programom tipično se koriste u kombinaciji s `lazarus` programom (Poglavlje 7), koji prikupljene podatke pokušava organizirati u smislene cjeline. Prilikom korištenja `unrm` alata potrebno je u obzir uzeti veličinu datotečnog sustava s kojeg se obnavljaju podaci kao i veličinu nezauzetog prostora. Također je vrlo važno da se prikupljeni podaci ne pohranjuju na isti datotečni sustav koji se analizira, budući da će to gotovo uvijek rezultirati prepisivanjem podataka koji se žele obnoviti. Veličina nezauzetog prostora posebno je važan parametar, budući da je istu količinu prostora potrebno osigurati na mediju na kojeg se vrši pohrana prikupljenih podataka.

Veličinu datotečnog sustav sa kojeg se obnavljaju podaci moguće je dobiti korištenjem `df` naredbe:

```
# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda1             12G   8.2G  2.8G  75% /
/dev/hda7             16G   8G    8G    50% /home
/dev/hda5             5.3G  2.7G  2.3G  54% /share
/dev/hda8             9.8G  7.8G  1.5G  84% /var
```

Iz dobivenog ispisa vrlo se jednostavno može vidjeti veličina pojedinog datotečnog sustava, kao i omjer zauzetog i slobodnog prostora. U nastavku je dan primjer korištenja `unrm` programskog paketa pokrenutog nad `/var` particijom. Izlaz programa preusmjeren je u `output` datoteku unutar `/home` datotečnog sustava kako bi se omogućila vjerna rekonstrukcija podataka iz nealociranog područja `/var` particije.

```
# unrm /dev/hda8 > /home/tct/output.unrm
```

Output datoteka dobivena pokretanjem `unrm` programa biti će kasnije korištena kao ulaz za `lazarus` program, koji će zabilježene blokove na disku pokušati organizirati u smislene cjeline. Nešto više riječi o načinu na koji `lazarus` program prepoznaje i organizira podatke na disku biti će u idućem poglavlju, gdje je detaljnije analiziran `lazarus` program i način njegovog korištenja.

Neke od opcija `unrm` programa, kojima je moguće utjecati na način njegova rada, opisane su u nastavku:

`-e` – kopiranje svih fizičkih blokova sa datotečnog sustava. Rezultat pokretanja programa je u ovom slučaju vrlo sličan rezultatu poznatog `dd` programa.

`-f <fstype>` - tip datotečnog sustava.

`-v` – *verbose* mode.

`device` – uređaj ili *image* datoteka koja se želi analizirati.

`start-stop` – Broj ili područje blokova na disku koji se žele analizirati.

`Unrm` trenutno podržava samo ext2fs datotečni sustav na Linux operacijskim sustavima i ufs datotečni sustav na Solaris i BSD operacijskim sustavima. Ovo se trenutno smatra nedostatkom `unrm` programa i u budućnosti se očekuje proširivanje mogućnosti programa.

7. Lazarus

Kako je već ranije spomenuto, `lazarus` programski paket namijenjen je obnavljanju izbrisanih podataka sa sustava, najčešće iz nealociranog područja na tvrdom disku, iako je moguće prikupljanje podataka i iz drugih izvora (radna memorija, *swap* i sl.). Program je pisan u Perl programskom jeziku. Iako `lazarus` može raditi na podacima iz različitih izvora (npr. *image* datoteka dobivena `dd` programom), program daje najbolje rezultate u kombinaciji sa `unrm` programom opisanom u prethodnom poglavlju (Poglavlje 6).

7.1. Način rada programa

U nastavku su opisani koraci u kojima program pokušava organizirati identificirane podatke:

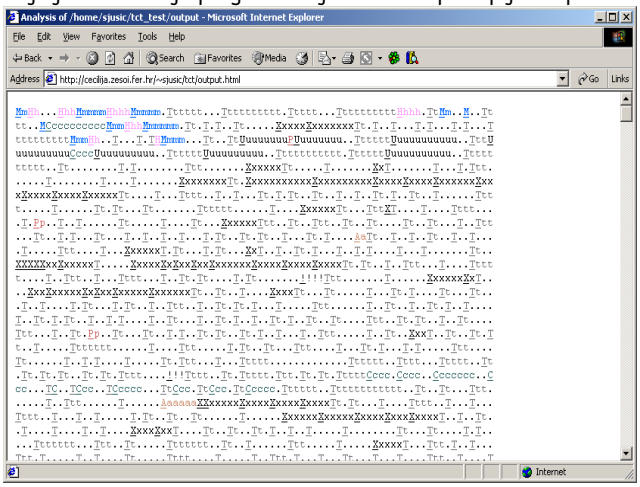
1. Program učitava blok podataka veličine određene parametrom `$BLOCK_SIZE` unutar konfiguracijske datoteke `lazarus.cf`.
2. Procjena sadržaja u pročitanom bloku (tekst ili binarna datoteka), kako bi se moglo nastaviti s daljnjom analizom podataka. Tip sadržaja određuje se pregledavanjem prvih 10% pročitanog sadržaja veličine $BLOCK_SIZE.
3. Ukoliko se radi o tekstualnom sadržaju, program pokušava utvrditi o kojem se tipu tekstualne datoteke radi (skripta ljuske, konfiguracijska datoteka i sl.).
4. Ukoliko se radi o binarnom sadržaju, izvršava se `file` naredba nad pročitanim nizom podataka.
5. Ukoliko je sadržaj unutar pročitanog bloka prepoznat, označava se kao blok određenog tipa. Ukoliko se prepoznati blok prema sadržaju razlikuje u odnosu na prethodno pročitani blok, pohranjuje se u novu datoteku, jer se smatra da se radi o različitim datotekama. Ukoliko je sadržaj unutar susjednih blokova identičan, tada se novi blok spaja sa prethodnim blokom, tj. smatra se da su oba bloka dio iste datoteke. Ukoliko novom bloku nije određen tip sadržaja isti se također nadovezuje na prethodno identificirani blok.
6. Rezultat izvršavanja programa sastoji se od dva dijela, identificirani blokovi podataka i mapa koja odgovara identificiranim blokovima podataka i koja olakšava njihovo pregledavanje. Identificirani blokovi podataka pohranjuju se u direktorij pod nazivom `blocks` (osim ako nije drukčije definirano opcijom `-D`). Imena datoteka u `blocks` direktoriju sadrže broj identificiranog bloka i pripadajuću oznaku (slovo) koja označava tip sadržaja u bloku (npr. slovo c označava datoteku s C programskim kodom, slovo H HTML datoteku, slovo `w password` datoteku, slovo l datoteke s log zapisima i sl.). Ove parametre moguće je modificirati uređivanjem `lazarus.cf` konfiguracijske datoteke. Svaka datoteka ujedno završava sa nastavkom `.txt`, kako bi se pojednostavilo pregledavanje datoteka Web preglednikom. Unutar konfiguracijske datoteke `lazarus` programa također su definirane i boje za svaki tip datoteke, kojima će pojedini tip sadržaja biti prikazan unutar HTML stranice.
   Mapa koja odgovara identificiranim blokovima sastoji se od niza znakova različitih boja, pri čemu svaka boja odgovara tipu podataka u pojedinom bloku. Slijed istovrsnih znakova podrazumijeva da je program sadržaj u susjednim blokovima ocijenio istovrsnim i da takvi blokovi pripadaju jednoj datoteci sustava. Prvi znak niza u tom slučaju je prikazan velikim slovom, čime se naglašava da se radi o prvom bloku određenog tipa podataka. Npr. niz `Cc` podrazumijeva da su tijekom analize identificirana dva susjedna bloka podataka čiji sadržaj odgovara C programskom kodu. Tim dvaju blokovima pridružena je jedna datoteka unutar `blocks` direktorija u kojoj se nalazi pripadajući sadržaj datoteke. Primjer jedne takve mape dobivene pokretanjem `lazarus` programskog paketa dan je u nastavku dokumenta (Slika 4).

7.2. Korištenje programa

U nastavku je dan primjer korištenja `lazarus` programskog paketa. Program je u svrhu primjera pokrenut nad `output.unrm` datotekom dobivenom korištenjem `unrm` programskog paketa, opisanog u prethodnom poglavlju (Poglavlje 6).

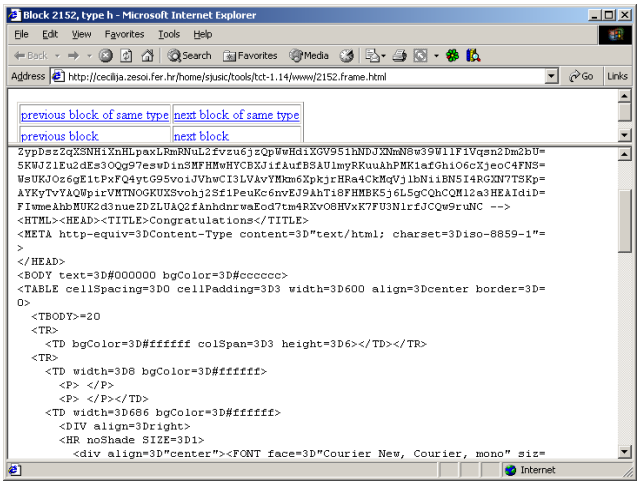
```
#lazarus -h /home/tct/output.unrm
```

Treba imati na umu da izvršavanje navedene naredbe može trajati duže vrijeme, ovisno o veličini ulazne datoteke. Opcija –h rezultirati će izlaznim datotekama u HTML, a ne ASCII formatu, što omogućuje pregledavanje dobivenih rezultata putem Web preglednika.
Izvršavanje navedene naredbe napraviti će dva direktorija `blocks` i `www`, u kojima se nalaze odgovarajuće html datoteke sa vezama na obnovljene datoteke. Na sljedećoj slici (Slika 4) prikazana je mapa koja omogućuje jednostavnije pregledavanje i analizu prikupljenih podataka.



Slika 4: Mapa `lazarus` programa koja olakšava pregledavanje i analizu prikupljenih podataka

Značenje dobivenog ispisa već je ranije opisano u poglavlju o načinu rada `lazarus` programa. Ukratko, svaki od znakova predstavlja jedan blok podataka, pri čemu boja određuje tip sadržaja datoteke. Niz istovrsnih znakova podrazumijeva niz blokova istovrsnog sadržaja. Prvi znak niza je ujedno i veza na HTML dokument u www direktoriju putem kojeg se ostvaruje pristup podacima u `blocks` direktoriju. Pritiskom na znak `H`, otvara se HTML datoteka sa obnovljenim sadržajem (Slika 5).



Slika 5: Obnovljena HTML datoteka

U gornjem dijelu prozora nalazi se sučelje za navigaciju koje omogućuje slijedno pregledavanje susjednih blokova. Ovakav način pregledavanja prihvatljiv je u slučajevima kada se želi steći globalna slika o prikupljenim podacima, no kada se želi analizirati točno određeni tip podataka onda se mogu primijeniti i druge metode (npr. korištenjem `grep` naredbe unutar `blocks` direktorija). U nastavku su opisane neke od osnovnih opcija lazarus programa:
`-B` – alternativna lokacija za `blocks` direktorij.
`-w` - alternativna lokacija za `www` direktorij.

-1 – program procesira svaki oktet podataka zasebno, a ne u blokovima.
-T – ne zapisuju se blokovi sa tekstualnim sadržajem.
-t – ne zapisuju se blokovi sa neprepoznatim sadržajem.

8. ILS

Ils program namijenjen je pregledavanju *inode* zapisa sa datotečnog sustava. Bez opcija program analizira samo *inode* zapise datoteka koje više ne postoje na sustavu. Prosljeđivanjem odgovarajućih opcija moguće je preciznije definirati način rada i koje se informacije žele prikupljati. Primjer korištenja programa dan je u nastavku:

```
# ./ils -r /dev/hda8 160000-170000
class|host|device|start_time
ils|cecilija.zesoi.fer.hr|/dev/hda8|1077727384

st_ino|st_alloc|st_uid|st_gid|st_mtime|st_atime|st_ctime|st_dtime|st_mode|
st_nlink|st_size|st_block0|st_block1
164186|f|0|0|1077619625|1077191728|1077619625|1077619625|100644|0|0|0
164193|f|501|0|1077418923|1077418923|1077418923|1077418923|100777|0|0|0|0
164212|f|501|0|1077418923|1077418921|1077418923|1077418923|100777|0|0|0|0
164220|f|501|0|1077418923|1077418920|1077418923|1077418923|100777|0|0|0|0
164234|f|501|0|1077418923|1077418921|1077418923|1077418923|100777|0|0|0|0
164235|f|501|0|1077619547|1077191729|1077619547|1077619547|100640|0|0|0|0
164236|f|501|0|1077418923|1077418921|1077418923|1077418923|100777|0|0|0|0
164239|f|501|0|1077619589|1077191727|1077619589|1077619589|100640|0|0|0|0
164242|f|501|0|1077418923|1077418922|1077418923|1077418923|100777|0|0|0|0
164244|f|501|0|1077418923|1077191729|1077418923|1077418923|100640|0|0|0|0
164254|f|501|0|1077418923|1077418922|1077418923|1077418923|100777|0|0|0|0
164528|f|501|0|1077619554|1077191729|1077619554|1077619554|100640|0|0|0|0
164531|f|501|0|1077619593|1077191727|1077619593|1077619593|100640|0|0|0|0
164561|f|501|0|1076814123|1077418923|1077418923|1077418923|100777|0|0|0|0
```

Kako se može primijetiti, ispis programa vrlo je sličan zapisima unutar body datoteke grave-robber programa, što ukazuje da grave-robber za prikupljanje informacija o *inode* zapisima koristi funkcionalnosti ils programa. Iako je većina polja slična kao i kod body datoteke, ipak postoje manje razlike u rezultatima. Kod ils programa prisutna su određena polja koja nisu prisutna unutar body datoteke:
-st_alloc – vrijednost f (*free*) znači da je *inode* struktura slobodna, dok vrijednost a (*allocation*) znači da je ista zauzeta.
-st_dtime – osim atime, ctime, i mtime parametara ils program sadrži i polje dtime koje predstavlja vrijeme proteklo od trenutka kada je datoteka predstavljena tom *inode* strukturom izbrisana.
Neke od osnovnih opcija programa opisane su u nastavku:
-e – program ispisuje sve *inode* zapise na zadanom datotečnom sustavu.
-f – tip datotečnog sustava.
-o – ispisuju se *inode* zapisi datoteka koje su izbrisane, ali koje su još uvijek otvorene ili koje se izvršavaju.
-a - ispisuju se samo zauzete *inode* strukture, one na koje pokazuje jedan ili više zapisa unutar direktorija.
-A – ispisuju se samo nezauzete *inode* strukture (one koje pripadaju datotekama koje više nisu prisutne na sustavu).
-z – ispisuju se *inode* strukture kod kojih ctime parametar sadrži vrijednost 0 (one *inode* strukture koje nikad nisu bile korištene).
-Z- ispisuju se *inode* strukture kod kojih ctime parametar sadrži vrijednost različitu od 0 (one strukture koje se trenutno koriste ili one koje pripadaju izbrisanim datotekama).
Podaci prikupljeni ils programom mogu poslužiti kao ulaz mactime programu za vremensku analizu podataka, ali prije toga moraju biti prilagođeni formatu koji mactime program očekuje. To je moguće postići korištenjem ils2mac programa, koji dolazi s TCT paketom, i koji se nalazi u extras direktoriju. Primjer korištenja ils i ils2mac programa dan je u nastavku:

```
# ./ils /dev/hda8 160000-170000 | ../extras/ils2mac >  ~tct/ils.output
```

Nakon toga se `ils.output` datoteka može iskoristiti kao `body` datoteka na temelju koje će `mactime` program raditi vremensku analizu.

```
# ./mactime -b ~tct/ils.output 12/12/2003
Feb 15 04 04:02:03   0 m.. -rwxrwxrwx nobody   root     <hda8-dead-164561>
Feb 19 04 12:55:27   0 .a. -rw-r----- nobody   root     <hda8-dead-164531>
                     0 .a. -rw-r----- nobody   root     <hda8-dead-164239>
Feb 19 04 12:55:28   0 .a. -rw-r--r-- root     root     <hda8-dead-164186>
Feb 19 04 12:55:29   0 .a. -rw-r----- nobody   root     <hda8-dead-164244>
                     0 .a. -rw-r----- nobody   root     <hda8-dead-164528>
                     0 .a. -rw-r----- nobody   root     <hda8-dead-164235>
Feb 22 04 04:02:00   0 .a. -rwxrwxrwx nobody   root     <hda8-dead-164220>
Feb 22 04 04:02:01   0 .a. -rwxrwxrwx nobody   root     <hda8-dead-164234>
                     0 .a. -rwxrwxrwx nobody   root     <hda8-dead-164236>
                     0 .a. -rwxrwxrwx nobody   root     <hda8-dead-164212>
Feb 22 04 04:02:02   0 .a. -rwxrwxrwx nobody   root     <hda8-dead-164242>
                     0 .a. -rwxrwxrwx nobody   root     <hda8-dead-164254>
Feb 22 04 04:02:03   0 m.c -rwxrwxrwx nobody   root     <hda8-dead-164212>
                     0 .ac -rwxrwxrwx nobody   root     <hda8-dead-164561>
                     0 m.c -rwxrwxrwx nobody   root     <hda8-dead-164242>
                     0 m.c -rwxrwxrwx nobody   root     <hda8-dead-164236>
                     0 m.c -rw-r----- nobody   root     <hda8-dead-164244>
                     0 m.c -rwxrwxrwx mtadic   root     <hda8-dead-164254>
                     0 m.c -rwxrwxrwx mtadic   root     <hda8-dead-164234>
                     0 m.c -rwxrwxrwx mtadic   root     <hda8-dead-164220>
```

9. ICAT

`Icat` program je vrlo praktičan alat koji omogućuje ispisivanje sadržaja datoteke prema zadanom broju *inode* strukture. Uz zadano ime uređaja i broj *inode* strukture, `icat` program ispisuje sadržaj blokova na disku vezanih uz navedenu *inode* strukturu. Npr., sljedeća naredba daje ispis datoteka u trenutnom direktoriju s brojem pripadajuće *inode* strukture (opcija `-i`):

```
# ls -ali
total 16
 853193 drwxrwxr-x    2 sjusic   sjusic       4096 Feb 26 12:30 .
 950811 drwx------   15 sjusic   sjusic       4096 Feb 26 12:29 ..
 853194 -rw-rw-r--    1 sjusic   sjusic         22 Feb 26 12:29 test1
 853197 -rw-rw-r--    1 sjusic   sjusic         22 Feb 26 12:29 test2
```

Zadavanjem sljedeće naredbe, moguće je doći do sadržaja pojedinih datoteka na temelju pripadajuće *inode* strukture.

```
# icat /dev/hda5 853197
Ovo je test2 datoteka.

Testiranje icat programa.

LSS & CERT.
```

`Icat` naredbi je u ovom slučaju proslijeđeno ime uređaja na kojem se nalazi odgovarajuća datoteka te broj *inode* strukture dobiven izvršavanjem `ls -ila` naredbe. Nakon brisanja određene datoteke, npr. `test2`, još uvijek je moguće do njenog sadržaja ukoliko je poznat broj *inode* strukture.

```
# rm test1
rm: remove regular file `test1'? y

# ls -ali
total 16
 853193 drwxrwxr-x    2 sjusic   sjusic       4096 Feb 26 12:30 .
 950811 drwx------   15 sjusic   sjusic       4096 Feb 26 12:29 ..
 853197 -rw-rw-r--    1 sjusic   sjusic         22 Feb 26 12:29 test2

# icat /dev/hda5 853194
Ovo je test1 datoteka.

Testiranje icat programa.

LSS & CERT.
```

10. PCAT

`Pcat` program omogućuje dolazak do podataka u radnoj memoriji odgovarajućeg procesa. S pokretanjem `pcat` naredbe treba biti posebno oprezan, budući da može uzrokovati probleme u radu sustava.

Sintaksa korištenja programa je vrlo jednostavna:

```
# pcat <opcije> PID
```

PID procesa čiji se dio radne memorije želi analizirati najjednostavnije je saznati korištenjem `ps` naredbe. Primjer korištenja programa dan je u nastavku.

```
# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   1388   248 ?        S    Feb24   0:04 init [5]
root         2  0.0  0.0      0     0 ?        SW   Feb24   0:00 [keventd]
root         3  0.0  0.0      0     0 ?        SW   Feb24   0:00 [kapmd]
root         5  0.0  0.0      0     0 ?        SW   Feb24   1:22 [kswapd]
root         6  0.0  0.0      0     0 ?        SW   Feb24   0:00 [bdflush]
root         7  0.0  0.0      0     0 ?        SW   Feb24   0:00 [kupdated]
root         8  0.0  0.0      0     0 ?        SW   Feb24   0:00 [khubd]
root        10  0.0  0.0      0     0 ?        SW   Feb24   0:09 [kjournald]
root       921  0.0  0.1   1464   276 ?        S    Feb24   0:00 syslogd -m 0
root       925  0.0  0.0   1388   248 ?        S    Feb24   0:00 klogd -x
rpcuser    962  0.0  0.0   1540   220 ?        S    Feb24   0:00 rpc.statd
root      1050  0.0  0.1   3520   500 ?        S    Feb24   0:02 /usr/sbin/sshd
root      1082  0.0  0.1   1436   276 ?        S    Feb24   0:00 crond
root      1143  0.0  0.0   1368   152 tty1     S    Feb24   0:00 /sbin/mingetty
tty1
root      1144  0.0  0.0   1368   152 tty2     S    Feb24   0:00 /sbin/mingetty
tty2
root      1145  0.0  0.0   1368   152 tty3     S    Feb24   0:00 /sbin/mingetty
tty3
root      1146  0.0  0.0   1368   152 tty4     S    Feb24   0:00 /sbin/mingetty
tty4
root      1147  0.0  0.0   1368   152 tty5     S    Feb24   0:00 /sbin/mingetty
tty5
root      1148  0.0  0.0   1368   152 tty6     S    Feb24   0:00 /sbin/mingetty
tty6
root      2937  0.0  0.6   6764 1680 ?        S    12:28   0:00 /usr/sbin/sshd
sjusic    2939  0.0  0.7   6904 1980 ?        S    12:28   0:01 /usr/sbin/sshd
sjusic    2940  0.0  0.5   4312 1408 pts/3    S    12:28   0:00 -bash
root      3271  0.0  0.2   2648  696 pts/3    R    14:58   0:00 ps aux
```

```
# pcat 1050 > sshd.dump
```

Budući da se radi o datoteci binarnog sadržaja, istu je potrebno pregledavati specijaliziranim alatima prilagođenim za tu svrhu, kao što je npr. `strings` program, binarni editor i sl.

11. Ostali programi

Osim dosad opisanih programa koji predstavljaju jezgru TCT programskog paketa, u paketu dolaze i drugi alati koji dodatno mogu pomoći u postupcima forenzičke analize. Dodatni alati nalaze se u `extras` direktoriju, a jedan od njih je već je ranije spomenut (`ils2mac`). Slijedi kratki opis:

- `bdf` – jednostavan program pisan u Perl programskom jeziku koji za zadanu tekstualnu ili binarnu datoteku traži ovisnosti o drugim datotekama. Ovisno da li se radi o binarnoj ili tekstualnoj datoteci program koristi odgovarajuće algoritme kojima se određuju ovisnosti o drugim datotekama. Primjer korištenja `bdf` programa dan je u nastavku:

```
# bdf -b ils
/home/sjusic/tct-1.09/bin/ils
/lib/ld-2.2.5.so
/etc/ld.so.cache
/usr/lib/libzvt.so.2.3.0
/usr/sbin/gnome-pty-helper
/var/log/wtmp
/var/log/lastlog
/usr/lib/libzvt.so.2.3.0
/usr/lib/libzvt-2.0.so.0.0.1
/usr/lib/libzvt-2.0/gnome-pty-helper
```

```
/lib/libz.so.1.1.4
/lib/libz.so.1.1.4
/lib/libz.so.1.1.4
/lib/libz.so.1.1.4
/usr/lib/libyafxplayer.so.0.0.0
                .
                .
```

- `entropy` - program za računanje entropije podataka, pri čemu se koristi Shannon-ova formula za izračun entropije. Rezultat izvršavanja `entropy` programa ima sljedeći format:

  ```
  <file> <file_entropy> <block_count> <min_entropy><max_entropy>
  ```

  a značenje pojedinih polja opisano je u nastavku:
  - `file` – ime datoteke za koju se radi izračun,
  - `file_entropy` – entropija,
  - `block_count` – broj analiziranih blokova podataka veličine block_size (veličinu bloka podataka na temelju koje se provodi analiza moguće je modificirati `-b` parametrom),
  - `min_entropy` – minimalna entropija,
  - `max_entropy` – maksimalna entropija.

  Primjer korištenja programa dan je u nastavku:

```
# entropy body
body   4.69    117     3.63    4.96
```

- `ils2mac` – program koji rezultate dobivene `ils` programom prilagođava `mactime` programu. Primjer korištenja `ils2mac` programa dan je u poglavlju u kojem je opisan `ils` program (Poglavlje 8).
- `findkeys` – program koji omogućuje pronalaženje kriptografskih ključeva unutar blokova podataka.

12. Zaključak

The Coroner's Toolkit programski paket danas je jedan od najpopularnijih alata za forenzičku analizu Linux/Unix operacijskih sustava. Program sadrži brojne alate koji korisniku omogućuju prikupljanje i analizu podataka sa kompromitiranih sustava, iako je potrebno odgovarajuće iskustvo za maksimalno iskorištavanje njegovih mogućnosti. U rukama iskusnih stručnjaka TCT predstavlja moćan alat i svakako se preporučuje svima koji se bave forenzičkom analizom. Dokumentacija koja dolazi sa programom prilično je potpuna i temeljita, što je također jedna od njegovih prednosti.