FACE: Automated Digital Evidence Discovery and Correlation

By

Andrew Case, Andrew Cristina, Lodovico Marziale,
Golden Richard, Vassil Roussev

Presented At
The Digital Forensic Research Conference
DFRWS 2008 USA  Baltimore, MD (Aug 11th - 13th)

DFRWS is dedicated to the sharing of knowledge and ideas about digital forensics research. Ever since it organized the first open workshop devoted to digital forensics in 2001, DFRWS continues to bring academics and practitioners together in an informal environment. As a non-profit, volunteer organization, DFRWS sponsors technical working groups, annual conferences and challenges to help drive the direction of research and development.

http://dfrws.org
FACE: Automated Digital Evidence Discovery and Correlation

Andrew Case, Andrew Cristina, Lodovico Marziale, Golden G. Richard III, Vassil Roussev

Me: PhD student and Research Assistant
Department of Computer Science
vico@cs.uno.edu
Forensics Challenge ‘08

- 3 Pieces of evidence
  - pcap network capture
  - Linux RAM dump
  - Files from a user home directory

- Object
  - Find user activity
  - Anything suspicious
  - Collaboration?

- Live forensics
The Problem

Disk has: filesystem, files, and MAC times

RAM has info on running processes, loaded modules, and live network connections.

Network capture has data in packets; part of logical network connection.

Plethora of tool exist for each types of data from each source.

But, the objects of interest (users, processes, network connections) are described in part by each. Investigator must “connect the dots.”

I am far too lazy to do this manually!
Solution

- **FACE**: Forensic Automated Correlation Engine
  - Correlate data for object across sources
  - Automate some of the “dot-connecting”
  - Penguin Power!

- **Good**: Have disk and network stuff
  - Wireshark …
  - Sleuthkit, Scalpel (shameless plug: new version soon!)

- **Bad**: linux RAM tools?
  - Idetect for 2.4
  - Crash and gdb
  - ???
Introducing: ramparser

Overview

- Linux 2.6, x86 (caveat later)
- C
- Processes
- Loaded modules
- Network connections
- More!

- Focus of this talk
Processes

- **The Plan**
  - Phase 1: Find `task_struct` for "init"
  - Phase 2: ?
  - Phase 3: Profit!

- **`task_struct`: the mother load**
  - System.map (`init_task`)
  - Carving
  - Caveat: `task_struct` representation
    - Kernel version
    - .config
    - CRAZY distro patches
Processes (more)

- Follow list or carve
- Basic: like `ps aux`
- Hardcore:
  - Code and data segments
  - Stack and heap
- Open files and network sockets: `/proc/<pid>/fd`
  - Linkage to disk image and network capture
- Mappings: `/proc/<pid>/maps`
  - Files
  - Libraries
  - Anonymous
Process Listing

```
#./ramparser challenge.mem -x
2152     501     501  gnome-session
/usr/bin/gnome-session

2262     501     501  bt-applet
bt-applet --sm-disable

2266     501     501  puplet
/usr/bin/python -tt /usr/bin/puplet

2269     501     501  vmware-user
/usr/lib/vmware-tools/bin32/vmware-user >/dev/null 2>&1 -blockFd 11

3048     501     501  firefox-bin
/usr/lib/firefox-1.5.0.12/firefox-bin
```
sock, socket and sk_buff, Oh my!

- Struct sock, struct socket
- Duplicate basic `netstat` functionality
- Buffers: struct `sk_buff`
  - More complete network capture
  - Sockets have per-connection
    - Send queue (more useful)
    - Receive queue (less useful)
    - Still in kernel (so not in network capture)
    - Exfiltration detection anyone?
## Netstat Output

执行命令`./ramparser challenge.mem -N`后的netstat输出如下：

<table>
<thead>
<tr>
<th>Proto</th>
<th>Local Address</th>
<th>Foreign Address</th>
<th>State</th>
<th>PID</th>
<th>Program name</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>0.0.0.0:111</td>
<td>0.0.0.0:0</td>
<td>LISTEN</td>
<td>1959</td>
<td>portmap</td>
</tr>
<tr>
<td>TCP</td>
<td>0.0.0.0:22</td>
<td>0.0.0.0:0</td>
<td>LISTEN</td>
<td>2311</td>
<td>sshd</td>
</tr>
<tr>
<td>TCP</td>
<td>0.0.0.0:60126</td>
<td>0.0.0.0:0</td>
<td>LISTEN</td>
<td>2332</td>
<td>rpc.statd</td>
</tr>
<tr>
<td>TCP</td>
<td>192.168.20.128:55071</td>
<td>192.168.20.129:80</td>
<td>ESTABLISHED</td>
<td>2521</td>
<td>firefox-bin</td>
</tr>
<tr>
<td>UDP</td>
<td>0.0.0.0:111</td>
<td>0.0.0.0:0</td>
<td></td>
<td>1959</td>
<td>portmap</td>
</tr>
<tr>
<td>UDP</td>
<td>0.0.0.0:32768</td>
<td>0.0.0.0:0</td>
<td></td>
<td>2332</td>
<td>rpc.statd</td>
</tr>
<tr>
<td>UNIX</td>
<td></td>
<td></td>
<td></td>
<td>2195</td>
<td>klogd</td>
</tr>
<tr>
<td>UNIX</td>
<td></td>
<td></td>
<td></td>
<td>2301</td>
<td>dhclient3</td>
</tr>
</tbody>
</table>
Module

- struct module
  - Carved or from list
  - Better carved, rootkits!

- Duplicates some `lsmod` functionality
### Module Listing

```
# ./ramparser challenge.mem -m

<table>
<thead>
<tr>
<th>Address</th>
<th>Module</th>
<th>Size</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xd154f100</td>
<td>uhci_hcd</td>
<td>25421</td>
<td>MODULE_STATE_LIVE</td>
</tr>
<tr>
<td>0xd1561880</td>
<td>ohci_hcd</td>
<td>23261</td>
<td>MODULE_STATE_LIVE</td>
</tr>
<tr>
<td>0xd156ee00</td>
<td>ehci_hcd</td>
<td>32845</td>
<td>MODULE_STATE_LIVE</td>
</tr>
<tr>
<td>0xd157cc00</td>
<td>mptspi</td>
<td>20041</td>
<td>MODULE_STATE_LIVE</td>
</tr>
<tr>
<td>0xd157f780</td>
<td>sd_mod</td>
<td>22977</td>
<td>MODULE_STATE_LIVE</td>
</tr>
<tr>
<td>0xd16a5a80</td>
<td>jbd</td>
<td>56553</td>
<td>MODULE_STATE_LIVE</td>
</tr>
<tr>
<td>0xd16c1a80</td>
<td>mptbase</td>
<td>52833</td>
<td>MODULE_STATE_LIVE</td>
</tr>
<tr>
<td>0xd170ae80</td>
<td>ext3</td>
<td>123081</td>
<td>MODULE_STATE_LIVE</td>
</tr>
<tr>
<td>0xd1735a00</td>
<td>scsi_mod</td>
<td>130637</td>
<td>MODULE_STATE_LIVE</td>
</tr>
<tr>
<td>0xd1805a00</td>
<td>sg</td>
<td>35933</td>
<td>MODULE_STATE_LIVE</td>
</tr>
</tbody>
</table>
```
Now what?

- Have overview of ram contents
- Disk <-> RAM <-> network linkage
- `ramparser` dump mode
- Feeds FACE correlation engine
  - Also simple network capture parsing
  - And some key config files
    - `/etc/passwd`
    - `/etc/group`
    - `/var/log/wtmp`
- Lookee here! (just a taste)
Process: ftp

PID: 2548

UID: root, 0

GID: root, 0

- Code: Hexdump Raw
- Data: Hexdump Raw
- Stack: Hexdump Raw
- Heap: Hexdump Raw

- Files:
  - FD: 0 /pts/0
  - FD: 1 /pts/0
  - FD: 2 /pts/0
  - FD: 3 socket:[6513]
  - FD: 4 socket:[6513]
  - FD: 5 socket:[6513]
  - FD: 6 /root/file2
- **Sockets:**

- **Mappings:**
  - /usr/bin/netkit-ftp
  - Anonymous
  - /usr/lib/gconv/gconv-modules.cache
  - /usr/lib/locale/locale-archive
  - /lib/tls/i686/cmov/libnss_nis-2.3.6.so
  - /lib/tls/i686/cmov/libnss-2.3.6.so
  - Anonymous
  - /lib/tls/i686/cmov/libnss_compat-2.3.6.so
  - /lib/tls/i686/cmov/libnss_files-2.3.6.so
  - Anonymous
  - /lib/tls/i686/cmov/libdl-2.3.6.so
  - /lib/tls/i686/cmov/libc-2.3.6.so
  - Anonymous
  - /lib/libcurses.so.5.5
User: root

UID: 0

GID: 0, root

Shell: /bin/bash

Home directory: /root

Last login: 2008-3-12T9:34:57

Processes belonging to user:

<table>
<thead>
<tr>
<th>Name</th>
<th>PID</th>
<th>Owner</th>
<th>Open Files</th>
<th>TCP/IP Sockets</th>
<th>Mappings</th>
</tr>
</thead>
<tbody>
<tr>
<td>init</td>
<td>1</td>
<td>root</td>
<td>1</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>migration/0</td>
<td>2</td>
<td>root</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ksoftirqd/0</td>
<td>3</td>
<td>root</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>watchdog/0</td>
<td>4</td>
<td>root</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>migration/1</td>
<td>5</td>
<td>root</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ksoftirqd/1</td>
<td>6</td>
<td>root</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### Files open:

- `/dev/initctl` opened by `init 1`
- `/dev/input/mice` opened by `Xorg 2417`
- `/dev/null` opened by `cron 2349`
- `/dev/null` opened by `cron 2349`
- `/dev/null` opened by `cron 2349`
- `/dev/null` opened by `sshd 2311`
- `/dev/null` opened by `sshd 2311`
- `/dev/null` opened by `sshd 2311`
- `/dev/null` opened by `dhclient3 2301`
- `/dev/null` opened by `dhclient3 2301`
Conclusions

- DFRWS challenge
  - I’m too lazy to connect all the dots
- ramparser:
  - processes including …
  - netstat and socket buffers
  - Modules
- FACE for automatic correlation
Future Work

- Really “present” work (half done)
- Generic 2.6 version
  - Dynamically build task_struct representation
  - Static symbols in System.map
  - From scratch
- Timestamp madness
- More RAM parsing fun
  - Block cache
  - Integrate swap
This is the end
Beautiful friend
This is the end
My only friend, the end
- JM

vico@cs.uno.edu