Hunting Linux Malware for Fun and Flags

Marc-Etienne M. Léveillé
Senior Malware Researcher
ESET
@marc_etienne_
About this presentation

- This presentation is an introduction to Linux malware analysis and incident response
  - Using commonly available tools
- There is a whole sandbox available where you can test your skills
  - This is where the fun and flags are!
  - Real-world scenarios
    - Trainees are granted root access
    - Use real (defused) malware
    - Real network interactions
You only need a web browser, an OpenVPN client and an SSH client to start hunting
Why this presentation and workshop?
Dissecting Linux/Moose
The Analysis of a Linux Rootkit-Based Worm Hungry for Social Networks
Olivier Bilodeau & Thomas Dupuy
May 2015

OPERATION WINDIGO
The revelation of a large Linux server-side backdoor stealing malware campaign

Unboxing Linux/Mumblehard
Muttering spam from your servers
Marc-Etienne M. Léveillé
April 2015

THE DARK SIDE OF THE FORSSHE
A landscape of OpenSSH backdoors
Why malware on Linux servers?

- Servers have a lot of bandwidth
- Servers have a high uptime
- As a result, they make good targets for
  - Sending spam
  - Reverse proxy
  - Open proxy
  - Traffic redirection
  - Hosting services (e.g. DNS) and web pages (e.g. phishing)
Why care?

- Bad IP reputation
  - Prevents sending legitimate email messages
- Slows down legitimate software and services
- Servers often host the most critical data in the enterprise
- Risk of data exfiltration
  - Passwords
  - E-mail addresses
  - Credit card numbers
  - Etc.

---

1 Jon Amato, Mario de Boer, *Gartner Technical Professional Advice – Solution Criteria for Endpoint Protection Platforms*, 2020-01-09
Why understand them?

- If you don’t find out how you were compromised, it might come back by the same door.
- If you don’t clean everything, it might come back using another backdoor.
- Understand what is at risk.
- Explain the behavior of very sneaky malware.
Incident response artifacts
Artifacts

● Filesystem
  – Logs
  – Malware persistence (if any)

● Memory
  – Process memory and state
  – Kernel memory

● Network
  – Configuration
  – Packet capture (in-band and out-of-band)
Common file metadata

- Name
- Size
- Type
  - Regular file
  - Directory
  - Symbolic link
  - Special (device)
- Owner
  - User
  - Group
- Access rights
  - Read, write and execute
  - Owner, group and others
- Timestamps
  - Access
  - Last modification
  - Last metadata modification
  - Creation date
Basic filesystem

- Finding new files
- `ls -alt | head`
  - List files that were recently modified in the current directory

```bash
# ls -lat | head
total 44
-rw------- 1 root root 4322 Oct 29 11:21 .bash_history
drwx------- 1 root root  22 Oct 28 23:52 .aptitude
-rw------- 1 root root  81 Oct 28 22:44 .lesshst
-rw------- 1 root root 5726 Oct 28 21:59 .viminfo
```
Basic filesystem

- **stat $FILE** – Full file details

```
# stat .viminfo
File: '.viminfo'
Size: 5726 Blocks: 16 IO Block: 4096
regular file
Device: 11h/17d Inode: 63052 Links: 1
Access: (0600/-rw-------) Uid: (0/root) Gid: (0/root)
Birth: -
```

All timestamps can be tampered with!
Basic filesystem

- `find / -newermt 2019-10-28`
  - Find files that were modified after October 28th
  - Based on the same metadata that can be tampered with

```
# find /home/james -newermt 2020-02-01
/home/james
/home/james/wwwroot/index.php
/home/james/wwwroot/static/css/plugins/isimg/css.php
/home/james/.lesshst
/home/james/.bash_history
```
Basic filesystem

- **file $FILE**
  - Identify file type

# file .viminfo
ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, stripped
Package integrity

- **debsums**
  - Dpkg-based distributions (Debian, Ubuntu)

- **rpm -Va**
  - RPM-based distributions (RHEL, CentOS, Fedora)
Malicious or not?

```bash
# rpm --verify keyutils-libs
(no error)
# rpm -qi keyutils-libs
Name : keyutils-libs                Relocations: (not relocatable)
Version : 1.4                      Vendor: CentOS
Release : 4.el6                    Build Date: Fri 22 Jun 2012 02:20:38
Install Date: Mon 27 Jan 2014 06:08:43 Build Host: c6b10.bsys.dev.centos.org
Group : System Environment/Base    Source RPM: keyutils-1.4-4.el6.src.rpm
Size : 59320                       License: GPLv2+ and LGPLv2+
Signature : RSA/SHA1, Sun 24 Jun 2012 06:18:51, Key ID 21efc4bf71fbfe7b
URL : http://people.redhat.com/~dhowells/keyutils/
Summary : Key utilities library
Description :
This package provides a wrapper library for the key management facility
system calls.
```
Logs

- `/var/log`
  - `auth.log`
  - HTTP logs
  - `messages`, `syslog`, etc.
- `systemd's journalctl`
- `auditd log`
Using auditd

- The Linux audit framework provides an auditing system that reliably collects information about any security-relevant (or non-security-relevant) event on a system.

- Part of the kernel
  - Must be enabled during kernel compilation or have a loadable kernel module
  - Enabled on most distributions

- Logs system calls and other types of events

- Logs can be sent over the network
Using auditd

- **auditctl**: Define what you want to log
- **ausearch**: Search in log files
  - Logs are text files, so `grep` and other tools work fine too for this task

```
# auditctl -a exit,always -S execve
[...]
# ausearch -m EXECVE
type=EXECVE msg=audit(1373838239.340:4474200): argc=4 a0="rm" a1="-f" a2="-f" a3="/tmp/q"
type=EXECVE msg=audit(1373838239.341:4474201): argc=4 a0="touch" a1="-r" a2="/etc/ssh/sshd_config" a3="/etc/ssh/ssh_config"
```
Offline filesystem

- If you don't have access to the live system but have an image of the partition.

- **Capture:** `dd if=/dev/sda3 of=$IMAGE_FILE`
  - Works over SSH too!

- **Browse:** `mount -o loop,ro $IMAGE_FILE /mnt`
System memory
# Analyzing a live process

- Identify running processes
  - `ps auxw`
  - `top, htop`

```
# ps auxw
root     7673  0.0  0.0  55164  5292 ?   Ss  Oct19  0:01 /usr/sbin/sshd -D  
root     7718  0.0  0.3 200676 127160 ?  Ss  Oct21  2:29 /lib/systemd/systemd  
root     7948  0.0  0.0 248844  25032 ?  Ss  Oct20  0:18 /usr/sbin/apache2 -k  
webuser  7953  0.0  0.0 141732  4188 ?   S   Oct20  0:06 /usr/sbin/apache2 -k  
webuser  5023  6.2  0.0  39764  8936 ?   Ss  Oct28 66:08 /tmp/.ICE-A5BF7  
[...]```
Analyzing a live process

- List open files and network streams
  - `ls/of -p $PID`

```
sshd 3642 sshd cwd  DIR   0,17  166  256 /   
sshd 3642 sshd rtd  DIR   0,17  166  256 /   
sshd 3642 sshd txt  REG   0,17 787080 2231 /usr/sbi... 
sshd 3642 sshd mem  REG   0,16   36265 /lib/x86... 
sshd 3642 sshd mem  REG   0,16  36267 /lib/x86... 
[...] sshd 3642 sshd 2u  CHR   1,3 0t0 1028 /dev/null
sshd 3642 sshd 3u  IPv4 146293912 0t0  TCP  158.69.117.51:ssh->182.100.67.59:41000 (ESTABLISHED)  
sshd 3642 sshd 4u  unix 0xffff88072c5cbc00 0t0 146289562 socket
```
● procfs provides a lot of useful details

● Mounted at /proc

● Contains one directory per process at /proc/$PID

$ ls /proc/3537
attr autogroup auxv cgroup clear_refs cmdline comm coredump_filter cpuset
cwd environ exe fd fdinfo gid_map io limits loginuid map_files maps mem
mountinfo mounts mountstats net ns numa_maps oom_adj
oom_score oom_score_adj pagemap patch_state personality projid_map root
sched schedstat sessionid setgroups smaps smaps_rollup stack
syscall task timers timerslack_ns uid_map wchan

stat statm status
procfs exe magic link

- Find the path of the executed file
  - `ls -l /proc/$PID/exe`

- Retrieve the executable file **even if it was deleted**
  - `cp /proc/$PID/exe malware.elf`

```
# ps aux | grep 25465
web  25465  6.6 0.0  39764  936 ? Ss Oct29 157:52 crond
# ls -l /proc/25465/exe
lrwxrwxrwx 1 web www-data 0 Oct 29 04:09 /proc/25465/exe -> /tmp/.ICE-684c
```
**procfs environ**

- `/proc/$PID/environ` contains the environment variables of a process separated by null bytes

```
# tr '\0' '\n' < /proc/1179/environ
MAIL_CONFIG=/etc/postfix
MAIL_LOGTAG=postfix
LANG=C
SSH_CONNECTION=10.0.2.2 58505 10.0.2.15 22
GENERATION=1654316
```
Process stalling

- Stop a process without destroying its resources.
  - `kill -SIGSTOP $PID`

- Resume a process previously stopped with SIGSTOP
  - `kill -SIGCONT $PID`
Process memory dump

- **Acquisition**
  - `gcore $PID` and `cp /proc/$PID/exe malware.elf`

- **Alternative acquisition tool**
  - `memfetch` from http://lcamtuf.coredump.cx

- **Analysis (simple)**
  - `strings`

- **Analysis (in-depth)**
  - `gdb malware.elf $PID.core`
Kernel memory

- **Acquisition**
  - VM snapshot
  - LiME (Linux Memory Extractor)

- **Analysis**
  - Volatility Framework

- **Only helpful if kernel is compromised via malicious kernel module (rootkit)**
Network
Network configuration

- Dump iptables rules
  - `iptables-save`
  - `ip6tables-save`

```
# iptables-save
[...]
-A POSTROUTING -s 0.0.0.0/0 --dport 8080 -o eth0 -j SNAT --to-source 89.4.205.9
-A PREROUTING -i eth0 -p tcp -m tcp --dport 8080 -j DNAT --to-destination 58.48.66.108:80
```
Network capture

- **Acquisition**
  - `tcpdump -i eth0 -s 0 -w capture.pcap`

- **Analysis**
  - `tshark -r capture.pcap`
  - Wireshark
  - `bro -r capture.pcap`
Malware analysis
Two approaches

- **Script-based malware**
  - PHP
  - Perl
  - Python

- **Compiled malware**
  - ELF executables
Script-based malware
Script-based malware

- Can be obfuscated
  - Removed whitespace
  - Variables renamed

```php
<?php function PXN1YnN0ci($a,$b){$c=array(139,164,40,72);if($b==62){$d=substr($a,$c[0]+$c[1],$c[2]);}elseif($b==12){$d=substr($a,$c[0],$c[1]);}elseif($b==92){$d=trim(substr($a,$c[0]+$c[1]+$c[2]));}return$d;} ?>
```
Most programming languages have a tool to *tidy* code
- Perl -> `perltidy`
- Python -> `PythonTidy`
- PHP -> `php-cs-fixer`
- Etc.

Rename variables with search and replace
Script-based malware

- Strings and literals can be packed
  - $43 \wedge 0x20 + 30$
  - $\text{x42}\text{x56}$

```php
$stg="ba"."\x73\x65"."64_d".\text{strrev("edo\x63e")};\text{eval($stg("JHN1cnZ1c191c2VyX2FnZWJWRVJbJ0hUVFBfVVNFU19BR0VOVCddOw0KJHN1cnZ1c19yZWZ1cmVyICAgICAgID0gQCRfU0VSVkVSWy[...]
dIVFRQX1JFRkVSRVInXTsNCiRzZXJ2ZXJfZm9yd2FyZGVkX2ZvciA9IEAkJ1N")}
```
Reversing script-based malware

- Always work in an isolated environment
- Use interactive prompts to evaluate parts of the code
  - Perl -> perl -de1
  - Python -> ipython
  - PHP -> php -a
  - Etc.
- Replace eval with print
Compiled malware
Compiled malware

- ELF executable in the native architecture of the system
- More challenging to understand
- Can also be packed
Reverse engineering compiled malware

- **Statically**
  - `strings`
  - `radare2`
  - `IDA Pro ($)`
Reverse engineering compiled malware

- Dynamically
  - `strace`
  - `ltrace` for dynamically linked binaries
  - `gdb`, or any other debugger you like
  - `gcore`

- Always work in an isolated environment when playing with malware
Apply what we’ve learned
This week you should

- Get hands-on experience using the Hunting Linux Malware for Fun and Flags workshop
Within three months you should

- Identify your Linux assets
  - Both on-premise and rented
  - Who has access to them?
  - From where are they reachable?

- Read articles and papers about Linux malware
Next you should

- Enable 2FA on all Linux servers
- Consider deploying security product (EPP and/or EDR) on your Linux servers
  
  - Malware and attacks impact systems, whether they are workstations, servers, mobile devices or assets in the cloud

---

1 Jon Amato, Mario de Boer, *Gartner Technical Professional Advice – Solution Criteria for Endpoint Protection Platforms*, 2020-01-09