Intrusion Detection using the Linux Audit Framework

Stephen Quinney <squinney@inf.ed.ac.uk>
School of Informatics
University of Edinburgh

“*the only secure computer is one that’s unplugged...*”

Two Distinct Goals

- Detection

https://burglaralarmbritain.wordpress.com

- Investigation

Understanding the Threat

- What are the likely privilege escalation routes?
- What are the likely aims of the attacker?
- How will an attacker try to hide/obfuscate a system compromise?

Root Kits

- Various tools and libraries – e.g. busybox
- Trojan versions of common applications and daemons (e.g. sshd).
- Often hidden in “plain sight” (e.g. /bin or /usr/bin)
- Covers tracks using kernel module – hides or obfuscates processes, directories, network traffic
- Acquires sensitive data via kernel module

Detection Requirements

- Reliable tools which cannot be subverted.
- Monitoring of important files for modification
  - preferably watching for both failure and success
- Monitoring of important resources (e.g. kernel)

Investigation Requirements

- Reliable log files which cannot be subverted.
- User authentication and session information.
- Record of **what** was changed.
- Record of **when** attempts are made to change files.
- Record of **who** attempted to change files.

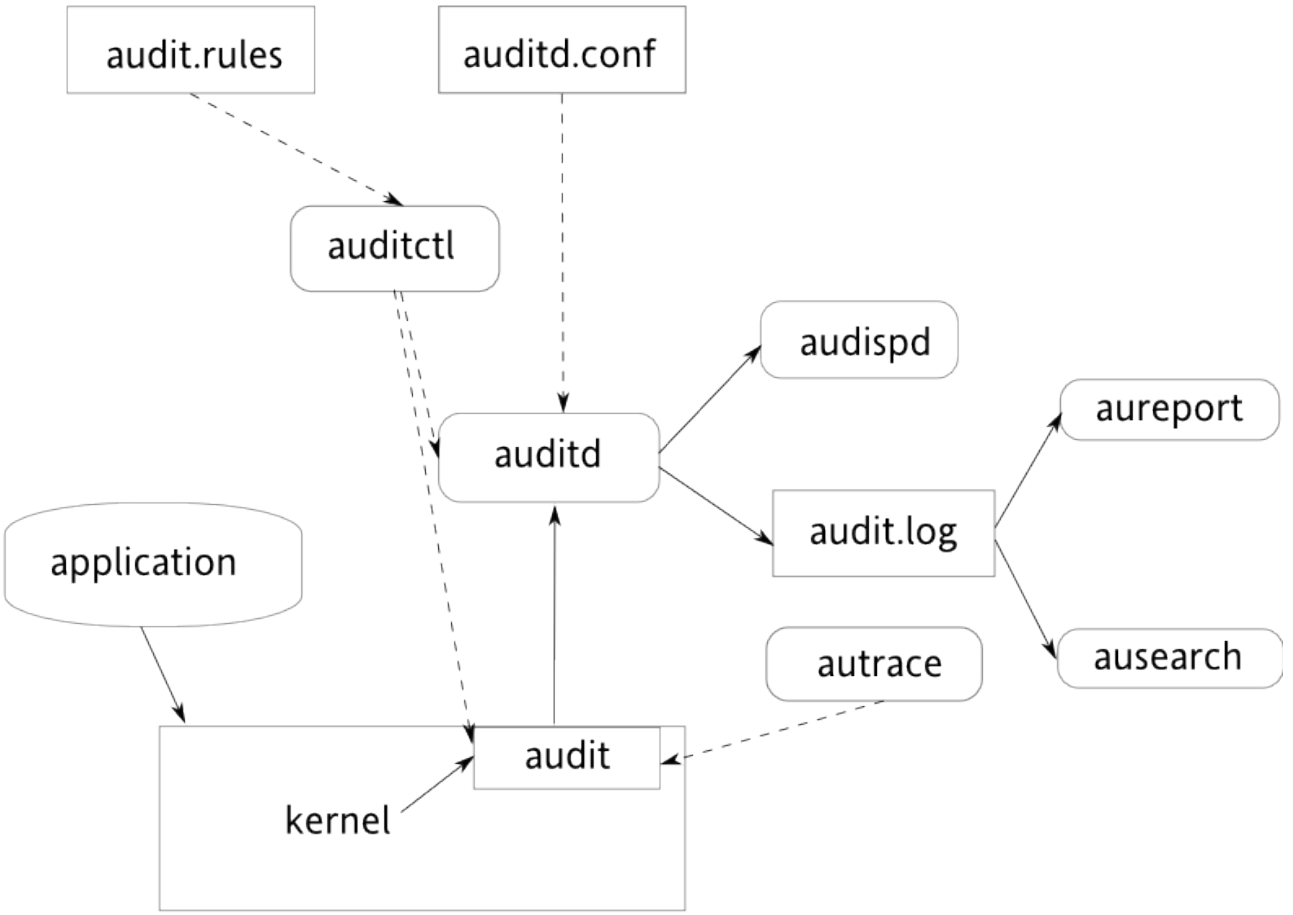
Intrusion Detection Strategies

- Regular filesystem scans – e.g. aide, rkhunter
  - Not immediate notification.
  - No record of when a change occurred.
  - No record of who made a change.
  - No record of failed attempts to change files.
  - Window of opportunity to hide signs of compromise.

Intrusion Detection Strategies

- Realtime monitoring (Linux Audit framework):
  - Can notify immediately on important events.
  - Records when a change occurs.
  - Records who made a change.
  - Can record failed attempts to change files.
  - No chance to hide rootkits to avoid detection.

Linux Audit Framework



https://www.suse.com/documentation/sles11/book_security/data/sec_audit_bigpicture.html

Documentation

- http://people.redhat.com/sgrubb/audit/
- https://www.suse.com/documentation/sles11/book_security/data/part_audit.html
- auditd.conf(5)
- auditd.rules(7)
- auditctl(8)
- auditd(8)
- aulast(8), aureport(8), ausearch(8),

Trustworthy Access Information

- Audit Daemon records authentication and session information.
- Each possible access point should use the PAM loginuid module (e.g. crond, login, gdm, sshd)
- The *loginuid* is immutable, even when using su and sudo which gives traceability.

```
session   required   pam_loginuid require_auditd
```

Listing logins

```
# aulast

squinney pts/0        brendel.inf.ed.a Wed Mar  4 18:15 - 18:42  (00:27)
squinney pts/5        sivori.inf.ed.ac Thu Mar  5 15:18 - 15:21  (00:02)
squinney pts/0        brendel.inf.ed.a Sat Mar  7 11:49 - 14:37  (02:48)
squinney pts/0        brendel.inf.ed.a Sun Mar  8 08:21 - 10:23  (02:01)
squinney pts/0        brendel.inf.ed.a Mon Mar  9 08:37 - 09:36  (00:59)
squinney pts/0        brendel.inf.ed.a Mon Mar  9 13:06 - 15:50  (02:43)
squinney pts/0        brendel.inf.ed.a Wed Mar 11 06:50 - 09:08  (02:17)
squinney pts/0        brendel.inf.ed.a Wed Mar 11 11:06 - 13:17  (02:10)
```

Listing logins

```
# aulast squinney --proof

squinney pts/0        brendel.inf.ed.a Wed Mar 11 11:06 - 13:17  (02:10)
    audit event proof serial numbers: 199424, 199430, 200596
    Session data can be found with this search:
    ausearch --start 11/03/15 11:06:55 --end 11/03/15 13:17:40 --session
32559
```

Searching for Users

```
# /sbin/ausearch --start 4/03/15 18:00 --end  11/03/15 14:00\
                 --loginuid squinney
```

Generate reports with aulast and aureport...

```
# /sbin/ausearch --start 4/03/15 18:00 --end  11/03/15 14:00\
                 --loginuid squinney --raw\
                 | aulast –stdin
```

```
# /sbin/ausearch --start 4/03/15 18:00 --end  11/03/15 14:00\
                 --loginuid squinney --raw\
                 | /sbin/aureport --login --interpret
```

Very useful!

Searching by Times

- *now* – right now
- *recent* – 10 minutes ago
- *today* – 1 second after midnight
- *yesterday* – 1 second after midnight on previous day
- *this-week* – 1 second after midnight on day zero of week
- *this-month* – 1 second after midnight on 1st day of month
- *this-year* – 1 second after midnight on 1st day of 1st month

- Alternatively specify exact date and/or time
- Beware locale-dependence!

Adding Your Own Rules

- Rules can be added using `auditctl`
- More typically done through `audit.rules` file
- Can build fairly complex rules to monitor files and syscalls.
- Can attach labels to recorded events for ease of searching.

Watching Files

- Can monitor files and directories for:
  - read (r)
  - write (w)
  - execution (x)
  - attribute changes (a)
- Rules for directories are applied recursively
- Cannot monitor root directory /

Simple File Monitoring

```
-w /bin   -p wa -k FS_mod
-w /boot  -p wa -k FS_mod
-w /etc   -p wa -k FS_mod
-w /lib   -p wa -k FS_mod
-w /lib64 -p wa -k FS_mod
-w /sbin  -p wa -k FS_mod
-w /usr   -p wa -k FS_mod
```

To search for any file modification records:

```
# /sbin/ausearch --key FS_mod
```

Simple File Monitoring

```
-w /bin/mount  -p x -k FS_suid
-w /bin/ping   -p x -k FS_suid
-w /bin/ping6  -p x -k FS_suid
-w /bin/su     -p x -k FS_suid
-w /bin/umount -p x -k FS_suid
```

Find all the setuid root programs in the / partition

```
# find / -mount -user 0 -perm -u=s,o=x
```

Monitoring syscalls

```
-a action,list -S syscall -F filter
```

- Actions: `never` or `always`
- Lists: `task, exit, user` or `exclude`
- Filter: `user, group, host, file` or `success` (for example)

Monitor the open and truncate syscalls for failures:

```
-a always,exit -S open -F success=0
-a always,exit -S truncate -F success=0
```

the syscalls can be combined to form a single rule:

```
-a always,exit -S open -S truncate -F success=0
```

or use multiple filters (restricts monitoring to /etc)

```
-a always,exit -S open -S truncate\
               -F dir=/etc -F success=0
```

Monitoring the Kernel

Watch the usage of various tools:

```
-w /sbin/insmod     -p x -k modules
-w /sbin/rmmod      -p x -k modules
-w /sbin/modprobe   -p x -k modules
```

Also monitor syscalls:

```
-a always,exit -F arch=b32 -S init_module\
               -S delete_module -k modules
-a always,exit -F arch=b64 -S init_module\
               -S delete_module -k modules
```

Note the two architectures needed on x86_64 machines

Reports

- Important filesystem changes
  - with whitelist. Maybe ignore packaged files? Maybe ignore changes initiated by root?
- Setuid root program usage
  - with whitelist
- Kernel changes
- Any reboots

Further Tips

- Enable auditing at boot-time.
  - Add `audit=1` to the kernel command line.
- Lock in the configuration.
  - `auditctl -e`
- Consider logging to a remote host.
  - see audispd(8)
- Look at standard rule sets – capp, lspp, nispom, stig are shipped with audit RPM.

Limitations

- Audit daemon can be overwhelmed
- Syscall rules are expensive, use them wisely!
- `ausearch:`
  - Awkward to use for complex searches
  - Output is difficult to parse
  - Consider using Python API
- Still not the whole story, use alongside other tools.

Summary

- The Linux Audit Framework is a great tool for detecting system intrusions!

- http://www.dice.inf.ed.ac.uk/publications/