Linux System Security Tunables

http://outflux.net/slides/2013/drupal/tunables.pdf

DrupalCon Portland 2013

Kees Cook <keescook@google.com>
(pronounced “Case”)

chrome

Who is this guy?

- Fun:
  - DefCon CTF
    - team won in 2006 & 2007
  - Debian
  - Ubuntu

- Jobs:
  - OSDL (proto Linux Foundation)
  - Canonical (Ubuntu Security)
  - Google (Chrome OS Security)

Overview

- Background
  - What do you mean, “post-intrusion”?
  - Layered defenses (aka “everything has bugs”)
- Best practices
  - Privilege separation (more than just root)
  - Kernel tunables (quick fixes)
- Start today

What do you mean, “post-intrusion”?

- It all started with a bug ...
  - … to gain remote execution
  - … to gain privilege escalation
  - … to gain kernel modification
  - … to gain more remote execution
  - rinse/repeat
- For example:
  - kernel.org penetration
  - http://www.crowdstrike.com/blog/http-iframe-injecting-linux-rootkit/
- Advanced Persistent Threat

Layered defenses

- Everything has bugs
- There is no such thing as “perfect security”
  - Any who thinks it exists hasn't had to deal with real-world attacks.
  - Best example: kernel bugs
    - Bypassing interfaces
    - Disregarding defenses because a bug “can't happen there”

Privilege separation

- Authentication hygiene (e.g. SSH keys)
- Discretionary Access Control (user-defined)
  - Separate users/roles
  - Strict permissions
- Mandatory Access Control (admin-defined)
  - AppArmor
  - SELinux
- Multi-factor authentication

Authentication hygiene

- Know where your credential storage lives
  - Keep away from devices with remote access
  - Store encrypted, tied to specific device

Authentication hygiene

```
$ hostname
local-device
$ ls ~/.ssh/id_*
id_rsa_device      id_rsa_device.pub
$ ssh-keygen -f ~/.ssh/id_rsa_foo -p
Enter new passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved with the new passphrase.
```

```
$ hostname
some-remote-system
$ ls ~/.ssh/id_*
ls: cannot access /home/kees/.ssh/id_*
$ cat ~/.ssh/authorized_keys
ssh-rsa A...== kees@phone
ssh-rsa A...wB kees@laptop
...
```

Authentication hygiene

- Actually check host keys

```
$ ssh-keygen -f /etc/ssh/ssh_host_rsa.pub -lv
1024 2b:29:a9:20:6f:9e:4a:de:b2:a3:b7:6b:31:bc:7f:f2
root@hostname (RSA)
+--[ RSA 1024]----+
|                 |
|                 |
|                 |
|         .       |
|  .. .   E       |
|   +o . o o      |
|  o oo + +       |
|+.=+. .. .       |
|=***So           |
+-----------------+
```

Discretionary Access Control

- Separate Unix users for:
  - Personal accounts
    - no direct access
  - Web services
    - cannot change execution
  - Service maintainers
    - no access to personal acct, limited system access
  - System admin
    - extremely powerful

Discretionary Access Control

- Pay attention to file system permissions
  - Clear lines between data and execution
- Control access via sudo or other keys

```
$ sudo cat /etc/sudoers
...
User_Alias    SOME_SERVICE = kees, gchaix, pholcomb
...
SOME_SERVICE  ALL = (some-maint) ALL


$ sudo cat ~some-maint/.ssh/authorized_keys
...
ssh-rsa AA...dF kees@laptop
ssh-rsa AA...e= gchaix@desktop
ssh-rsa AA...J1 pholcomb@phone
```

Mandatory Access Control

- Specify precisely what access the service has.
  - AppArmor
  - SELinux
  - SMACK
  - Tomoyo

Mandatory Access Control

- AppArmor profile “hats” with Apache
  - http://wiki.apparmor.net/index.php/Mod_apparmor_example
  - /etc/apparmor.d/usr.lib/apache2.mpm-prefork.apache2

```
$ cat /etc/apparmor.d/apache2.d/spaces.org
^spaces.org {
  #include <abstractions/apache2-common>
  #include <abstractions/base>
  #include <abstractions/php5>

  /srv/www/spaces.org/{html,private}/    r,
  /srv/www/spaces.org/{html,private}/**  r,
  owner /srv/www/spaces.org/private/**   wkl,
  /home/jcook/scripts/spaces.hits        r,

  /srv/www/spaces.org/logs/*             w,
}
```

Multi-factor authentication

- Downside of sudo: 1 password for 2 accounts
- Add a physical token
  - HID
  - RSA token
  - yubi-key
  - google-authenticator
  - duo-unix
    - https://www.duosecurity.com/pricing

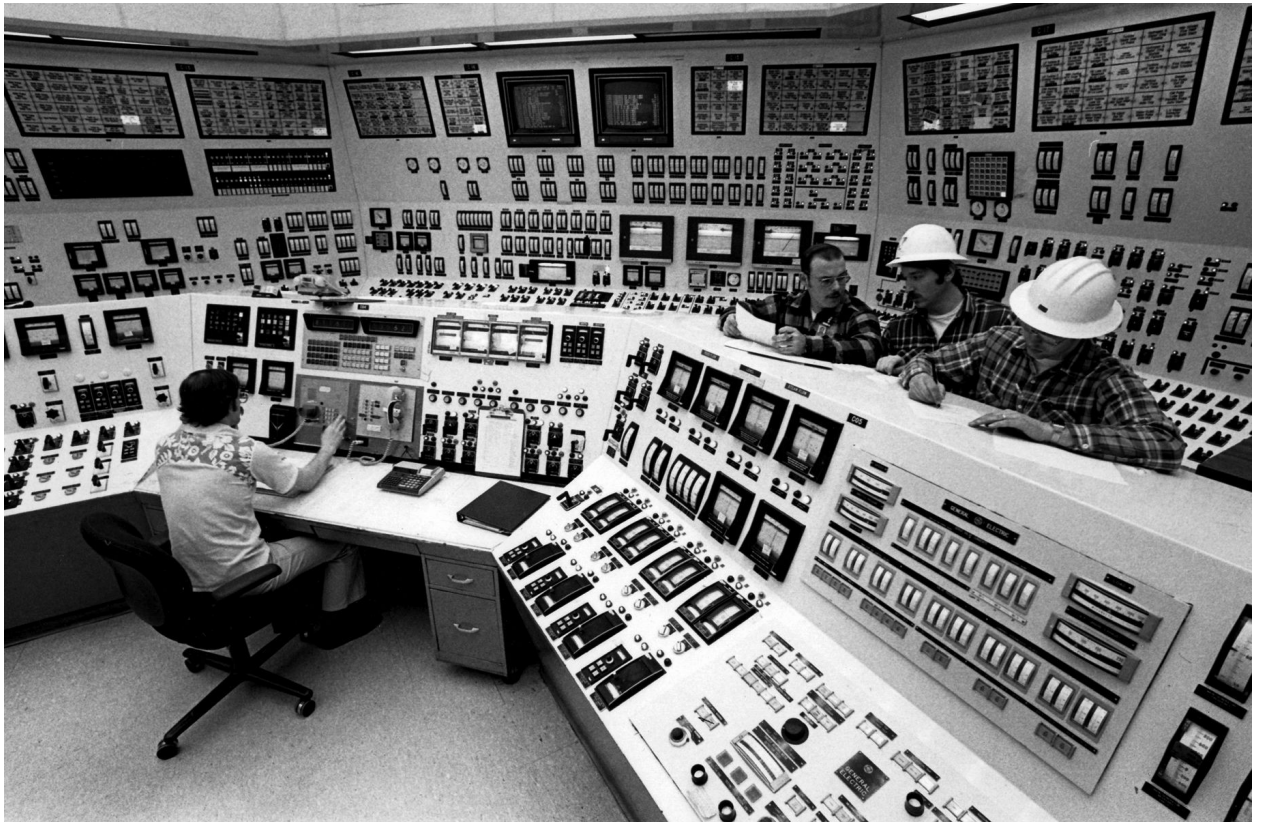
Multi-factor authentication

- PAM with duo-unix

```
$ sudo apt-get -y install libpam-duo
...
$ sudo vi /etc/security/pam_duo.conf
... ikey = ... skey = ...
$ sudo pam-auth-update
...
$ sudo -K
$ sudo -s
[sudo] password for kees:
Duo two-factor login for kees

Enter a passcode or select one of the following options:

 1. Phone call to XXX-XXX-5694
 2. SMS passcodes to XXX-XXX-5694 (next code starts with: J)

Passcode or option (1-2):
```

Kernel tunables

- Network
  - tcp_syncookies
- Debug
  - perf_event_paranoid
  - ptrace_scope
  - kptr_restrict
  - dmesg_restrict
- Virtual Memory
  - mmap_min_addr
- Filesystem
  - protected_hardlinks
  - protected_symlinks
- Kernel Execution
  - modules_disabled



Kernel tunables

- Tree of items in /proc/sys/
- Configure either directly or via “sysctl” tool
- Boot-time configured from /etc/sysctl.d
- Documented in kernel source (and a bit in man-pages)
  - Documentation/sysctl/

```
$ find /proc/sys -type f | wc -l
1272
$ cat /proc/sys/kernel/randomize_va_space
2
$ sysctl kernel.randomize_va_space
2
$ sudo sysctl kernel.randomize_va_space=2
kernel.randomize_va_space = 2
```

net.ipv4.tcp_syncookies=1

- Encodes connection details in TCP options
- Self-regulating
- Downside is loss of options that don't matter

kernel.yama.ptrace_scope=1

- Block “sibling” processes from modifying each other
  - SSH hijacking
- Disrupts attach (“strace -p”, “gdb -p”) but not debugging of launched child processes
- Could also get crazy and use higher modes:
  - 2: root only (CAP_SYS_PTRACE)
  - 3: nothing can use ptrace

vm.mmap_min_addr=65536

```
$ cat runme.c
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>


int main(void) {
    struct stat *info = NULL;



    printf("%ld\n", info->st_ino);
    return 0;
}

$ make runme
$ ./runme
Segmentation fault (core dumped)
```

vm.mmap_min_addr=65536

```
$ cat runme.c
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <sys/mman.h>

int main(void) {
    struct stat *info = NULL;

    mmap(0, 4096, PROT_WRITE,
         MAP_FIXED | MAP_PRIVATE | MAP_ANONYMOUS, -1, 0);
    printf("%ld\n", info->st_ino);
    return 0;
}

$ make runme
$ ./runme
0
```

kernel.kptr_restrict=1

- Kernel addresses are useful to attackers

```
$ grep tcp_transport /proc/kallsyms
ffffffffa045b180 d xs_tcp_transport     [sunrpc]
ffffffffa045b1e0 d xs_bc_tcp_transport  [sunrpc]

$ sudo grep ^nfsv3 /proc/modules
nfsv3 34322 1 - Live 0xffffffffa0582000 (F)
```

kernel.kptr_restrict=1

- Kernel addresses are useful to attackers

```
$ grep tcp_transport /proc/kallsyms
0000000000000000 d xs_tcp_transport     [sunrpc]
0000000000000000 d xs_bc_tcp_transport  [sunrpc]

$ sudo grep ^nfsv3 /proc/modules
nfsv3 34322 1 - Live 0x0000000000000000 (F)
```

kernel.dmesg_restrict=1

- So much handy information for an attacker

```
$ dmesg
dmesg: klogctl failed: Operation not permitted
```

fs.protected_symlinks=1

- Classic Time-of-Check-vs-Time-of-Use attack
  - “/tmp symlink attack”

```
$ cd /tmp
$ ln -s /etc/cron.d/evil predictable-filename
$ readlink predictable-filename
/etc/cron.d/evil

#!/bin/bash
if [ ! -e /tmp/predictable-filename ]; then
    echo “eeek” >/tmp/predictable-filename
fi

# /the/buggy/script

$ cat /etc/cron.d/evil
eeek
```

fs.protected_symlinks=1

- Classic Time-of-Check-vs-Time-of-Use attack
  - “/tmp symlink attack”

```
$ cd /tmp
$ ln -s /etc/cron.d/evil predictable-filename
$ readlink predictable-filename
/etc/cron.d/evil

#!/bin/bash
if [ ! -e /tmp/predictable-filename ]; then
    echo “eeek” >/tmp/predictable-filename
fi

# /the/buggy/script
cat: predictable-filename: Permission denied
```

fs.protected_hardlinks=1

- Principle of “least surprise”

```
$ df -h /etc /var/tmp
Filesystem                      Size  Used Avail Use% Mounted on
/dev/mapper/sysvg-rootlv         30G   22G  6.8G  76% /
/dev/mapper/sysvg-rootlv         30G   22G  6.8G  76% /

$ ls -l /etc/shadow
-r--r----- 1 root shadow 3213 Apr 28 22:08 /etc/shadow
$ cd /var/tmp
$ ln /etc/shadow evilness
```

fs.protected_hardlinks=1

- Principle of “least surprise”

```
$ df -h /etc /var/tmp
Filesystem                      Size  Used Avail Use% Mounted on
/dev/mapper/sysvg-rootlv         30G   22G  6.8G  76% /
/dev/mapper/sysvg-rootlv         30G   22G  6.8G  76% /

$ ls -l /etc/shadow
-r--r----- 1 root shadow 3213 Apr 28 22:08 /etc/shadow
$ cd /var/tmp
$ ln /etc/shadow evilness
$ ls -l evilness
-r--r----- 1 root shadow 3213 Apr 28 22:08 evilness
```

fs.protected_hardlinks=1

- Principle of “least surprise”

```
$ df -h /etc /var/tmp
Filesystem                      Size  Used Avail Use% Mounted on
/dev/mapper/sysvg-rootlv         30G   22G  6.8G  76% /
/dev/mapper/sysvg-rootlv         30G   22G  6.8G  76% /

$ ls -l /etc/shadow
-r--r----- 1 root shadow 3213 Apr 28 22:08 /etc/shadow
$ cd /var/tmp
$ ln /etc/shadow evilness
ln: failed to create hard link ‘evilness’ => ‘/etc/shadow’:
Operation not permitted
```

kernel.modules_disabled=1

- Trivial jump from root user into kernel code
  - Remember the iframe injector?
- Most servers have relatively static hardware
  - Just be sure to preload what you might need

kernel.modules_disabled=1

```
$ cat /etc/modprobe.d/disable.conf
# To disable module loading after boot, "modprobe disable"
# can be used to set the sysctl that controls module loading.
install disable /sbin/sysctl kernel.modules_disabled=1

$ tail -n2 /etc/rc.local
modprobe disable
exit 0

$ cat /etc/modules
hid
usbhid
usb-storage
disable
```

Start today

- Make a plan
- Prioritize the changes
  - Stop logging in as root over telnet before you configure Mandatory Access Controls
- Make the changes
- Automate verification
  - Cacti? Nagios? Cron? Anything!

Questions?

http://outflux.net/slides/2013/drupal/tunables.pdf

keescook@{google.com,chromium.org}
kees@{outflux.net,debian.org,ubuntu.com}