COMP 3704 Computer Security

Christian Grothoff
christian@grothoff.org

http://grothoff.org/christian/

Operating System Security

OS security is important:

- OS is more trusted than any application
- OS is responsible for resource allocation
- OS is tiny compared to applications (hope!)
- OS can improve application security (PAX!)

OS Security Functions

- Resource allocation (memory, disk, bandwidth, CPU)
- Enhanced application security (VM, PAX, RBAC)
- Local access control (data, devices)
- Network access control ($\Rightarrow$ Wednesday!)

Resource Allocation

- `int setrlimit(int resource, const struct rlimit * rlim)`
- RLIMIT_AS, RLIMIT_CORE, RLIMIT_CPU, RLIMIT_RSS, RLIMIT_NOFILE, ...
- `man bash` – look for ulimit
- `man 2 nice`; `man 1 nice`
- File system quotas (see link on webpage)

Virtual Machines

- Similar to operating system
- Can provide more fine-grained protections (for example, guard private fields from other parts of application)
- Can be used to achieve stronger isolation than what ordinary OSes provide
- Should generally be treated like an OS in security analysis

UNIX File Permissions

- Standard permissions: Read (4), Write (2), eXecute (1)
- Differentiation by: User, Group, Others
- `man chmod`, `man chown`
- Default permissions are $arg\&\ mask$ where `arg` is specified by the application. For `mask`, see `man umask`

Process User Identifiers

- Each process is associated with multiple user IDs: real, effective, saved and possibly others
- Real UID is the UID of the process that created this process. Can only be changed if effective UID is root (0).
- Effective UID is used for permission checks; EUID can be changed to real UID or to saved UID. If EUID is 0, anything goes.
- New files are created using the effective UID

SUID, SGID

- If permissions of executable file are set to SUID, SUID of executed process will be set to UID of the file's owner.
- This allows the program to switch to those permissions using `seteuid(SUID)`
- Processes also have multiple group IDs, the same rules apply.
- Binaries with SUID and SGID can be used to elevate permissions

Groups

- Each user can be in any number of groups
- `newgrp` can be used to change the current group ID
- `/etc/group` specifies group memberships
- `groups` lists current memberships

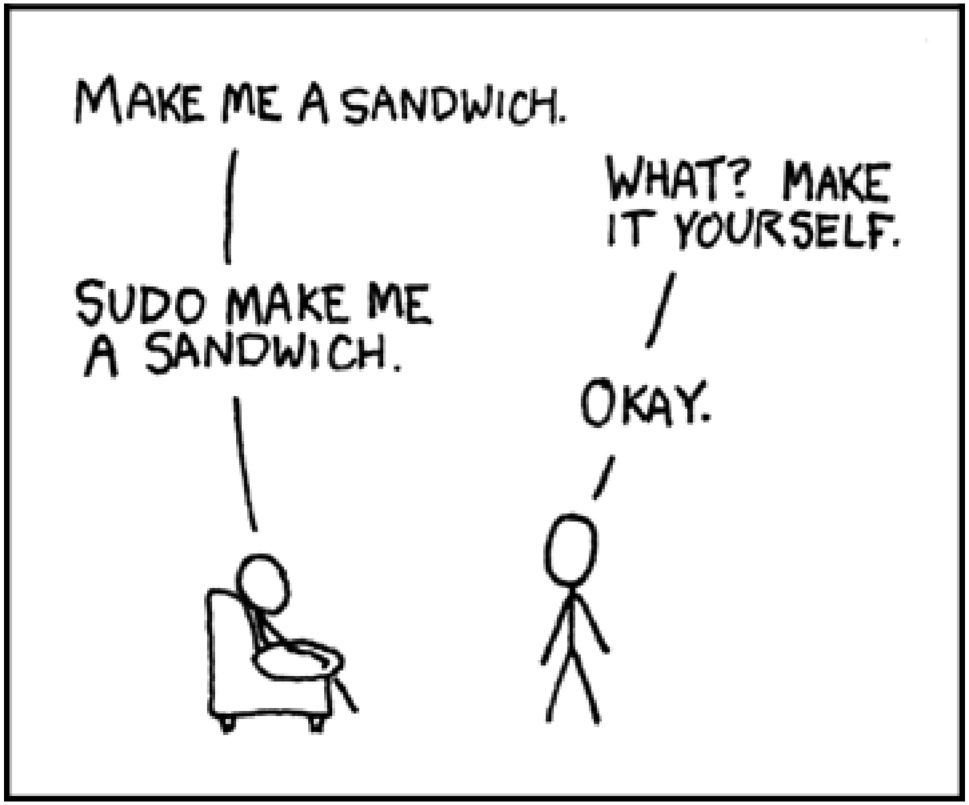
PAM: Pluggable Authentication Modules

Flexible mechanism for authenticating users:

- NIS / LDAP / SQL / SMB
- `/etc/passwd`, `/etc/shadow`
- USB dongle

PAM also specifies policies, such as `root`'s password unlocking ordinary users's X locks (or not).

SUDO


MAKE ME A SANDWICH.
WHAT? MAKE IT YOURSELF.
SUDO MAKE ME A SANDWICH.
OKAY.

SUDO

- Allow ordinary users to run certain commands as root
- Logs commands and command options
- `/etc/sudoers` configures `sudo`

chroot: Go to Jail!

- chroot changes the root directory ("/") to the specified path

⇒ Process only sees limited portion of the file system

- chroot requires root priviledges
- root-owned processes can escape the jail – remove CAP_SYS_CHROOT!
- man chroot

Role-based Access Control

- `root` is usually god
- RBAC can be used to restrict `root`
- RBAC specifies for every user/group/process which <u>specific</u> system calls are allowed
- Some RBAC systems can even be used to specify call sequences
- The kernel itself is still trusted!

Kernel Modules

- Allow drivers to be loaded on-demand
- Avoid useless code to be present in stock kernel

⇒ Less code, better security

- Problem: `root` can load malicious code into the kernel

⇒ Use RBAC to limit loading or monolithic kernels without support for modules

BSD Security Levels

- `sysctl -w kern.securelevel=N` for $N \in \{0, 1, 2\}$
- 0: default (insecure mode)
- 1: some restrictions (such as no IO to raw devices, no module loading)
- 2: no mounting of disks with write permissions
- 3: no changes to firewall configuration possible
- The security level cannot be lowered other than by rebooting the system

Hardware to the Rescue

- Read-only memory
- Non-executable heap (Harvard machine!)
- Non-executable stacks
- BIOS password (and boot-loader password!)
- Disk password

Crypto Loop Devices

```
# dd if=/dev/urandom of=/dev/sda1
# losetup -e aes-256 /dev/loop0 /dev/sda1
# mkfs.ext3 /dev/loop0
# mount /dev/loop0 /home
# umount /home
# losetup -d /dev/loop0
```

Patch Tuesday

- Why Tuesdays?
- `apt-get update`
- `apt-get upgrade`

CERT/CC Intruder Detection Checklist

1. Examine log files
2. Look for setuid and setgid files
3. Check system binaries
4. Check for packet sniffers
5. Examine files run by 'cron' and 'at'
6. Check for unauthorized services
7. Examine `/etc/passwd` file
8. Check system and network configuration
9. Look everywhere for unusual or hidden files
10. Examine all machines on the local network

Debian Paranoia Ideas

- CD distribution
- Disable kernel modules
- Logging through serial cable (or LP)
- `chattr +i` for `/bin, /sbin`, etc.
- Create a honeypot (to learn about intrusion)

Questions

?

Problem

Suppose `root` gave somebody `sudo` rights to the `cat` command.

What can the user do with this?

Problem

Suppose Nathan had exploited his SUDO `apt-get` priviledges to obtain `root` on `linux1.cs.du.edu`.

How can Andrei find out?