Laurea Magistrale in Cybersecurity
Dipartimento di Informatica
Sapienza Università di Roma

Hacking Unix

Local Access

Lecturer: Daniele Friolo
Slides: Fabio De Gaspari

Local Access

From local to root: *Privilege Escalation*

1. Attack the system
   - Weak passwords, symlink, kernel flows, …
2. Exploit System misconfiguration
   - File/directory permissions, SUID, ...

Remaining persistent: Advanced Persistent Threats (APT)

1. Trojan
2. Rootkits
3. Log cleaners

Local Access

From local to root: *Privilege Escalation*

1. Attack the system
   - Weak passwords, symlink, kernel flaws, …
2. Exploit System misconfiguration
   - File/directory permissions, SUID, …

Remaining persistent: Advanced Persistent Threats (APT)

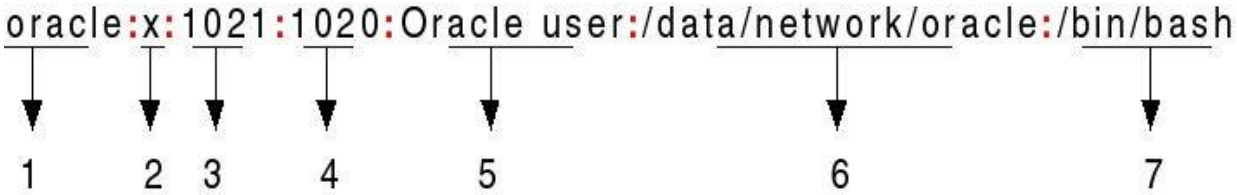
1. Trojan
2. Rootkits
3. Log cleaners

Weak Passwords

Weak password are a never-ending security issue

- Users are lazy, given the chance they will always take path of least resistance
- Password requirements help only marginally.
  - Consider P@ssword1: >8 characters, symbols, capitals, numbers. Cracks in probably <10 seconds
- Dictionary attacks allow to quickly crack easy-to-guess passwords

Weak Passwords: /etc/passwd



1: Username
2: Password stub, real password stored in /etc/shadow 3, 4: User id and Group id
5: User info such as full name, phone, …
6: Absolute path of home directory
7: Absolute path of default shell

Weak Passwords: /etc/shadow

1: Username

2: Hashed password in Modular Crypt Format (MCF):

- $hash_id$salt$hash$

3: Last password change

4: Number of days until password change is allowed

5: Number of days before password change is required

6: Number of days before password expires to warn user

7: Number of days after password expiration after which account is disabled. 8: Expiration date of account (days since epoch)

Weak Passwords: Meet John the Ripper

Great tool to crack hashes

- Automatically handles MCF format
- Easy to configure and extend
- Good set of mangling rules
  - Can be extended with separate input files

```
john --wordlist=<dictionary> <password file>
```

Weak Password Countermeasures

You guessed it, require stronger password

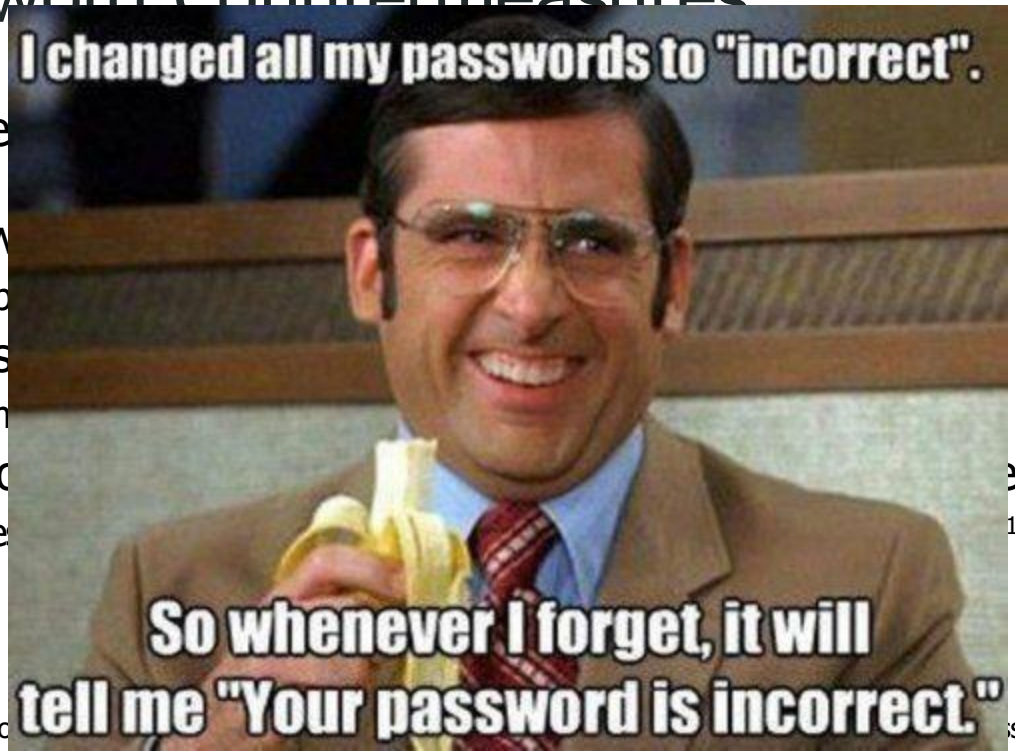
- Longer passwords?
  - Users will probably just add the same letter repeated at the end
- Regex checks for long repetitions?
  - Users will most likely add easy to remember characters. Birthday?
- Enforcing strong passwords is hard, if users don't collaborate
  - Passphrases can potentially help, but even those can be easily defeated[1]

1. https://arstechnica.com/information-technology/2012/03/passphrases-only-marginally-more-secure-than-passwords-because-of-poor-choices

Weak Password Countermeasures

You guessed it, re

- Longer passw
  - Users will p
- Regex checks
  - Users will n
- Enforcing stro
  - Passphrase



1. https://arstechnica.com/info swords-because-of-poor-choices

Symlink

Symbolic links are pointers from one file to another

- Completely transparent to applications and users
  - Writing to a file "temp", or to a symlink to temp makes no difference from the user's PoV
- Can be exploited by attackers to trick a program into referencing other files during execution
- Creating symlinks to protected files does not requires permissions
  - `ln -s /etc/shadow ./link_to_shadow` allowed as unprivileged user

Symlink Countermeasures

Secure coding practices are the best countermeasures

- Use O_EXCL | O_CREAT when using open syscall
  - Fails if files already exists, with error EEXIST
- Beware of reading/writing from publicly accessible directories
  - More fun on this later…
- File system mount options can also help
  - cat /proc/mounts

Race Condition

Programs are more interesting when they run with elevated privileges

- Generally, many programs elevate privileges only temporarily
  - Short time window for attackers to exploit: race condition
- An attacker exploiting a program in this window *wins the race*
- One common example is signal-handling race conditions

Signal-Handling Race Conditions

Signals are a Unix mechanism used to notify processes about particular events

- Allow handling such events asynchronously
  - e.g., ctrl+z sends SIGTSTP to all foreground processes
- Signals *alter the execution flow* of running processes
- Attackers can exploit signals to halt processes while they have elevated privileges
  - Or in general, try to alter the program flow to make it easier to exploit

Signal-Handling Countermeasures

Not much you can do as a user, secure code practices are key

- As always, keep third party programs up to date

Core File Manipulation

Core files are a dump of the address space of a process created when it exits unexpectedly

- Invaluable tool to debug programs
- If not managed properly, can expose sensitive information to attackers
  - Including password hashes from /etc/shadow
- Web application crashes can generate core dumps in the root directory
  - www.vulnerablesite.com/core thank you for your core file
  - A small survey[1] shows approximately .1% Alexa Top 1M sites are affected

1. https://blog.hboeck.de/archives/887-Dont-leave-Coredumps-on-Web-Servers.html

Core Files Countermeasures

Manage your core files properly

- Disable core files if you can
  - Very inconvenient for sysadmin and developers
- Configure core dumps with appropriate permissions
  - Limit who can access the dumps
  - General rule of thumb, if you're storing dumps where anyone can download them, you might want to revise your security practices…

Shared Libraries

Shared libraries (DLL in Windows) allow multiple programs to call routines from a common library upon execution.

- Allow to save memory and makes it easier to maintain code
  - Updating the shared library effectively updates all program using it
- Also makes for a great target for attackers
  - Injecting altered library allows attackers to execute arbitrary code on the system
  - Compromise multiple programs at once

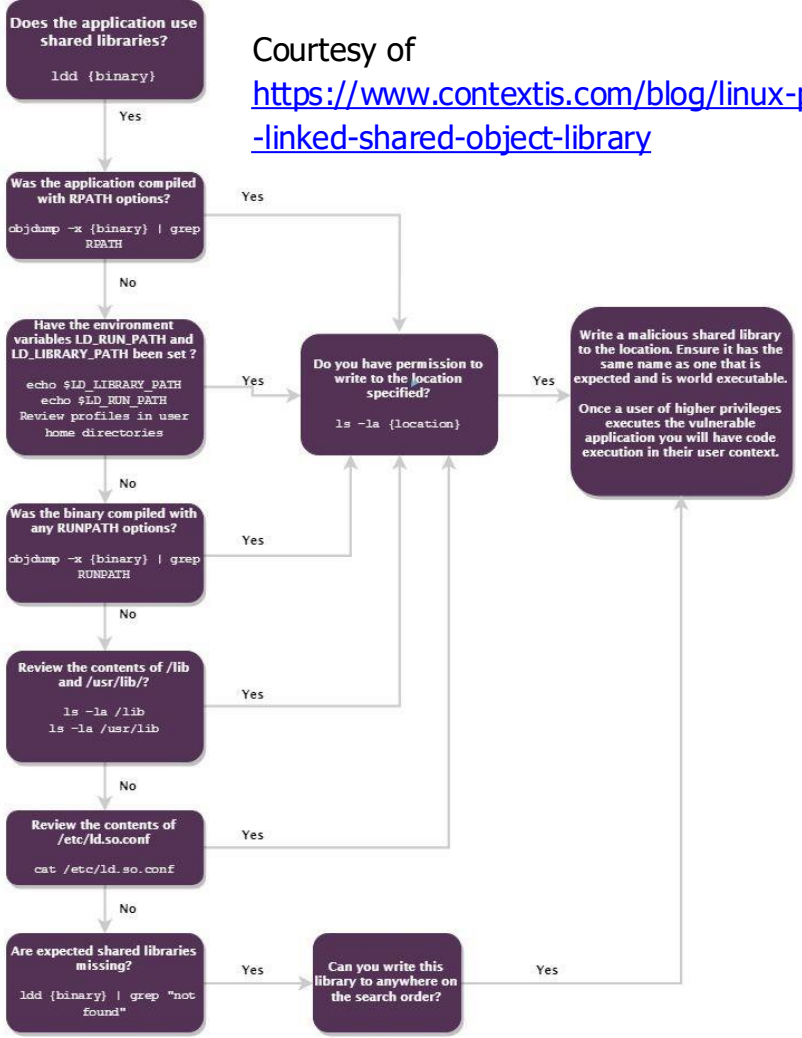
Shared Libraries

Where does the system look for shared libraries? (see https://linux.die.net/man/1/ld)

- Any directories specified by rpath-link options (only effective at link time)
- Any directories specified by –rpath options (directories included in the executable and used at runtime)
- LD_RUN_PATH environment variable (link time resolution)
- LD_LIBRARY_PATH environment variable (run time resolution)
- Directories in the DT_RUNPATH or DT_RPATH. (DT_RPATH ignored if DT_RUNPATH exist)
- /lib and /usr/lib
- Directories listed in /etc/ld.so.conf

Shared Libraries

Courtesy of
https://www.contextis.com/blog/linux-privilege-escalation-via-dynamically-linked-shared-object-library



Shared Libraries Countermeasures

- As always, follow security practices
- Use -rpath when linking whenever possible
- Make sure LD_RUN_PATH, LD_LIBRARY_PATH, RUNPATH don’t include directories with weak permission settings

Kernel Flaws

Unix system are highly complex and robust

- However, with high complexity comes higher chance of introducing bugs
- The kernel is responsible for enforcing the overall system's security model
  - Including, permissions, escalation of privileges, signals handling, …
  - Kernel security bugs compromise the security model at the root
- Attackers can easily exploit kernel flaws to obtain complete control of the system

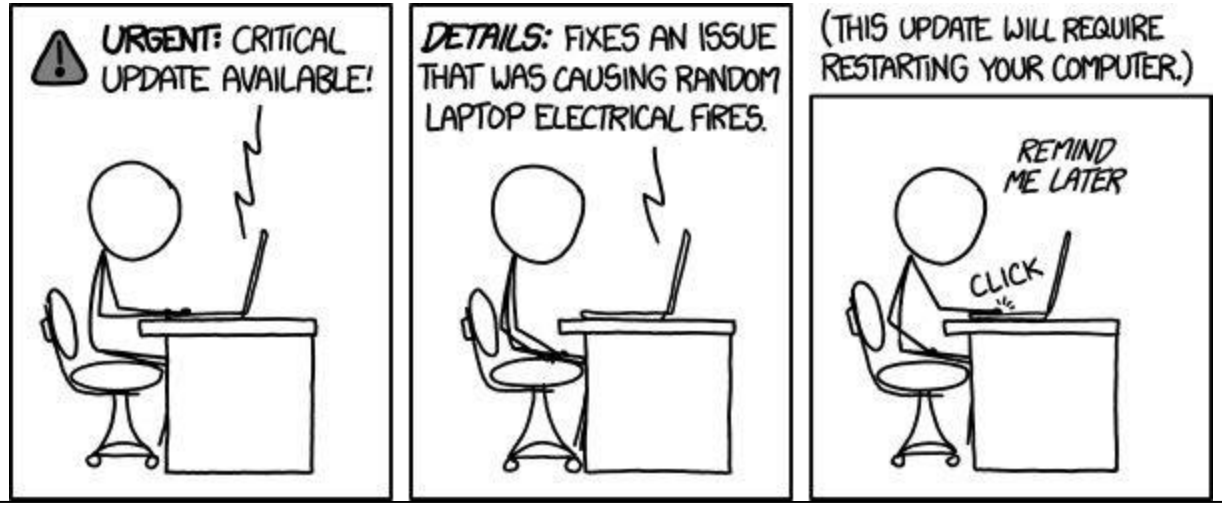
Kernel Flaws

Unix sys

- How
- The
  -
  -
- Atta

syst

Always keep your kernel up to date with security patches



Local Access

From local to root: *Privilege Escalation*

1. Attack the system
   - Weak passwords, symlink, kernel flaws, …
2. Exploit System misconfiguration
   - File/directory permissions, SUID, …

Remaining persistent: Advanced Persistent Threats (APT)

1. Trojan
2. Rootkits
3. Log cleaners

Unix Permissions

File permissions are specified by 3 access classes: *user*, *group* and *others*

- user class permissions apply to the owner of the file
- group class permissions apply to users who are part of a specific group
- others class permissions apply to everyone else

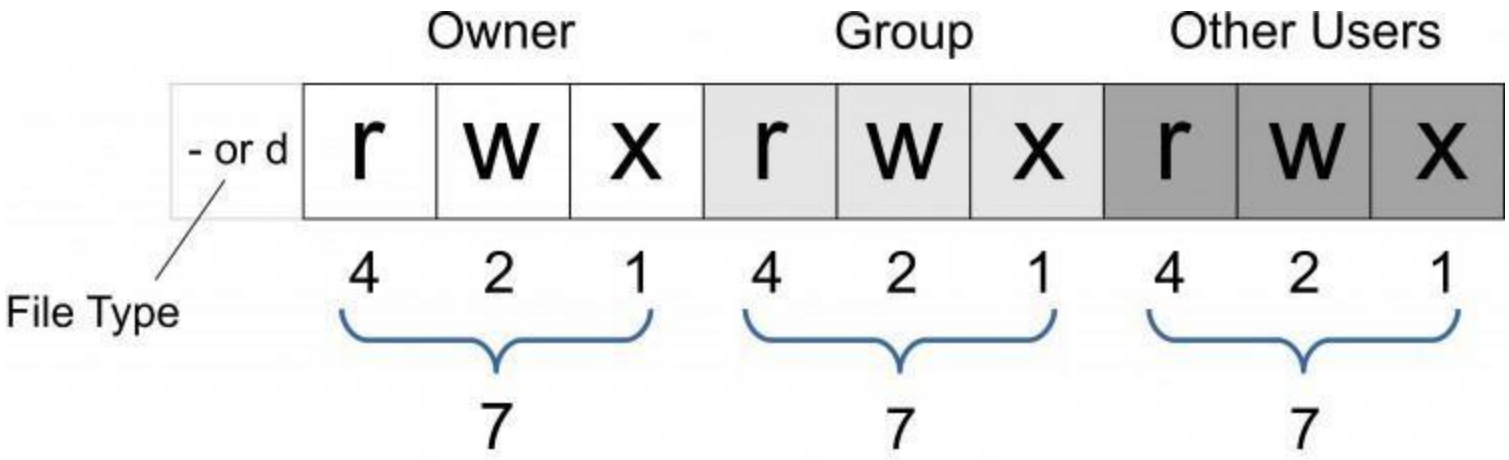
Unix Permissions

For each access class, 3 access types (*modes*) can be set:

- read (r): defines if the given class can read the file
- write (w): defines if the given class can write the file
- execute (x): defines if the given class can execute the file

Unix Permissions

ls -l
<filename>



Unix Permissions

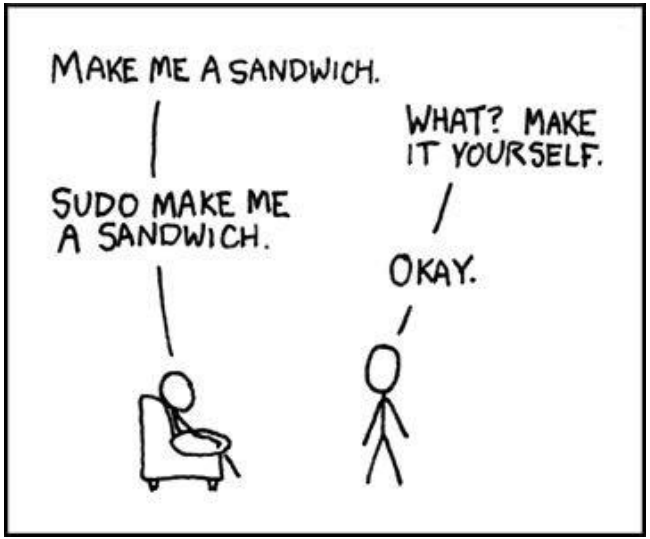
Each file also has 3 *special modes*, valid for all classes

- Set user id (SUID)
- Set group id (SGID)
- Sticky

SUID and Security Implications

When a file with SUID is executed, the process assumes the effective user ID of the owner of the file

- Provides flexibility and allows for temporary elevation of privileges
  - *sudo, passwd* require SUID to work
- Executing a SUID file owned by root spawns a process with EUID 0 (root)
  - What could go wrong?



Exploiting misconfigured SUID

- Many SUID programs create temp files, stored in /tmp
  - $ stat /tmp: Access: (1777/drwxrwxrwt)
  - Who uses /tmp? *strings /bin * | grep tmp*
  - What if we add a bunch of symlinks in here?
- Poorly coded SUID programs
  - buffer overflows are even more fun with suid

SUID Countermeasures

The best countermeasure is to remove as many SUID files as possible

- Create an inventory of SUID files
- Remove SUID bit wherever possible
- Use nosuid option of mount (check cat /proc/mounts)

World-Writable Files

Files that can be modified by any user

- Generally used for convenience/laziness
- Can (and should) be avoided most of the time

Let's see what we find in our VM:

```
find / -not -type l -not -path "/proc/*" -perm -o+w 2>
/dev/null
```

(ignores symlinks and /proc)

World-Writable Countermeasures

- Do not use world-writable files, unless you really really *really* need to
- Create an inventory and review all world-writable files
- *Never* leave startup scripts as world writable

Local Access

From local to root: *Privilege Escalation*

1. Attack the system
   - Weak passwords, symlink, kernel flows, …
2. Exploit System misconfiguration
   - File/directory permissions, SUID, ...

Remaining persistent: Advanced Persistent Threats (APT)

1. Trojan
2. Rootkits
3. Log cleaners

APT - Trojans

Trojans are malware hidden in otherwise normal files

- Once an attacker has root, he can attach a trojan to any file/program in the system
  - e.g., corrupted version of login/ssh that stores username/passwords typed by users
- Trojans can easily open backdoors that allowing remote connections
  - Or to circumvent firewall restrictions, reverse connections from the target to the attacker's machine
- The sky's the limit…

Trojan Countermeasures

Hard to detect without proper setup

- Creation/modification date can be spoofed, size can be basically identical
- Hash of the binary is the best way to verify the legitimacy
  - Requires a database storing hash for every program in your system
- Cannot rely on backup copies after compromise, as the backup could also be compromised

APT - Sniffers

There is lots of interesting and confidential information in network traffic

- Sniffers allow attackers to "sniff" all network traffic that goes through the same local network segment
  - Exacerbated by the increasing use of wifi networks
- Attackers can get a good understanding of potential target services, while remaining completely passive
- If traffic is not properly secured, attackers can also obtain critical information such as passwords and confidential documents

Sniffers Countermeasures

Sniffers are borderline impossible to detect, but their effectiveness can be limited

- Use switched network topologies, limiting traffic that each host can see
  - Still vulnerable to arp-spoofing attacks
- Sniffer detection
  - Sniffer need the NIC to be in promiscuous mode and tend to create big log files
  - Root attackers can however take steps to compromise detection attempts
- Always use network-level encryption
  - TLS and IPSec can effectively protect traffic from sniffers
  - Not applicable in all scenarios

APT - Log Cleaning

System logs contain information regarding all activities going on in the system

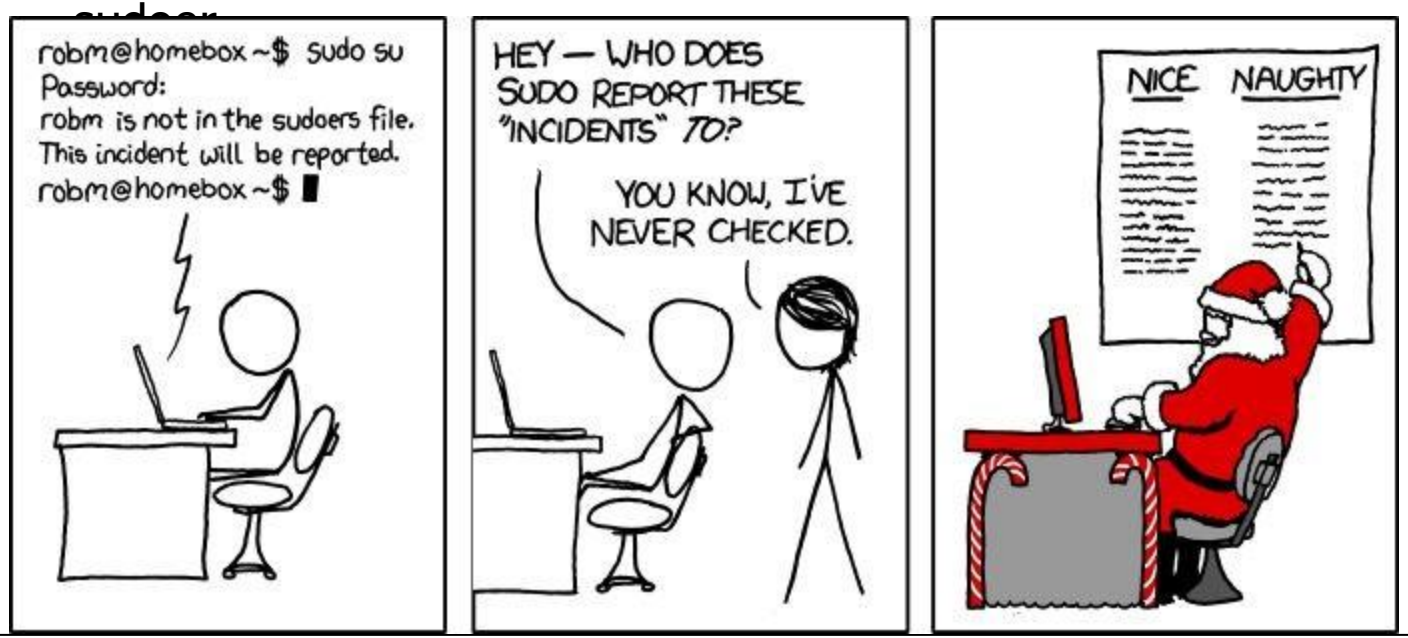
- Including the activity of the attacker
- Log cleaners are part of virtually every rootkit, allowing attackers to delete their traces from your system
  - e.g., login records, shell command history
- Log cleaners can also intercept programs that send log files to remote servers
  - ptrace() is a debugging function that allows to trace and control the execution of other processes (think gdb)

APT - Log Cleaning

Sys

-
-
-
-

For instance, when you try to sudo and you're not a



rs

cat /etc/syslog.conf

APT - Kernel Rootkits

The worst type of rootkits, used to compromise the kernel of the system itself

- Allows attackers to compromise all system programs, without modifying any of them
- Many venue of attack for kernel rootkits
  - Loadable kernel modules has historically been abused to plant rootkits
  - Advanced methods based on raw memory modifications (/dev/mem)

APT - Kernel Rootkits

Several approaches for kernel rootkits to intercept system calls

- Most obvious one is modifying system call table, redirecting calls to custom routines
  - Easily detectable through integrity check
- Corrupt system call handler, to point to attacker's system call table
  - Does not modify original system call table, bypassing integrity checks
  - Can be achieved by exploiting /dev/mem
- Hack interrupt descriptor table or interrupt handler
  - Logically similar to changing system call table or system call handler

Kernel Rootkit Countermeasures

You can’t trust any binaries on your system, nor the system itself

- Certain tools can freeze the status of every process and capture key information
  - Can also extract image of running processes
- Prevention is really the only countermeasure when it come to kernel rootkits
  - Even worse if we consider hardware rootkits…

Conclusion

System security is a very hard and never ending battle. It’s also a lot of fun though.