

Sicurezza e privacy lab

Linux ACL and Password cracking

Dott. Matteo Zoia

SPLab@di.unimi.it



Reviewing the basics

Linux ACL

The Linux filesystem gives us three types of permissions. Here is a simplified review:

- User (or user owner)
- Group (or owner group)
- Other (everyone else)

With these permissions, we can grant three types of access:

- Read
- Write
- eXecute

These levels of access are often adequate in many cases. Say that you have a directory where files from the accounting department live. You might set these permissions to:

```
drwxr-xr-x 14 accounting accounting 4096 Oct 19 05:25 accounting
```

Reviewing the basics, example

Linux ACL

```
drwxrwxr-x 14 accounting accounting 4096 Oct 19 05:25 accounting
```

The accounting service user (the user owner) can read and write to the directory, and members of the accounting group (or owner group) can read and write, but no one else can.

```
drwxrwx--- 14 accounting accounting 4096 Oct 19 05:25 accounting
```

What if you have an **intern (Chloe)** who needs to be able to read certain files? Or maybe people in the sales **department also need access to the accounting** owner's files to create invoices in order to bill customers, but you don't want them to see the other reports that his team generates. This type of situation is what Linux Access Control Lists (ACLs) were intended to resolve.

getfacl, viewing current ACL

Linux ACL

ACLs allow us to apply a **more specific set of permissions** to a file or directory without (necessarily) changing the base ownership and permissions. We can view the current ACL using the **getfacl** command:

```
kali@kali:~/Desktop/lab2/example$ getfacl accounting/  
# file: accounting/  
# owner: kali  
# group: kali  
user::rwx  
group::r-x  
other::r-x
```

We can see that right now, there are no ACLs on this directory. In this case, we expected this result, since I just created this directory in the lab and haven't done anything other than assigning ownership. So, let's start by adding a default ACL.

ACL vs Capabilities

Linux ACL

ACL Entry				
Capabilities Entry		X's medical record	Y's medical record	Z's medical Record
	Alice (GP)	r,w,x	r	-
	Bob (GP)	-	r, w, x	-
	Charlie (Physician)	r,w	r,w	r,w
	Dean (Professor)	r,w,x	r,w,x	r,w,x

setfacl, setting an ACL

Linux ACL

The syntax for setting an ACL looks like this:

```
setfacl [option] [action/specification] file
```

The 'action' would be -m (modify) or -x (remove), and the specification would be the user or group followed by the permissions we want to set. In this case, we would use the option -d (defaults). So, to set the default ACL for this directory, we would execute:

```
kali@kali:/home$ setfacl -d -m user:rwX /accounting
```

getfacl, effective and mask

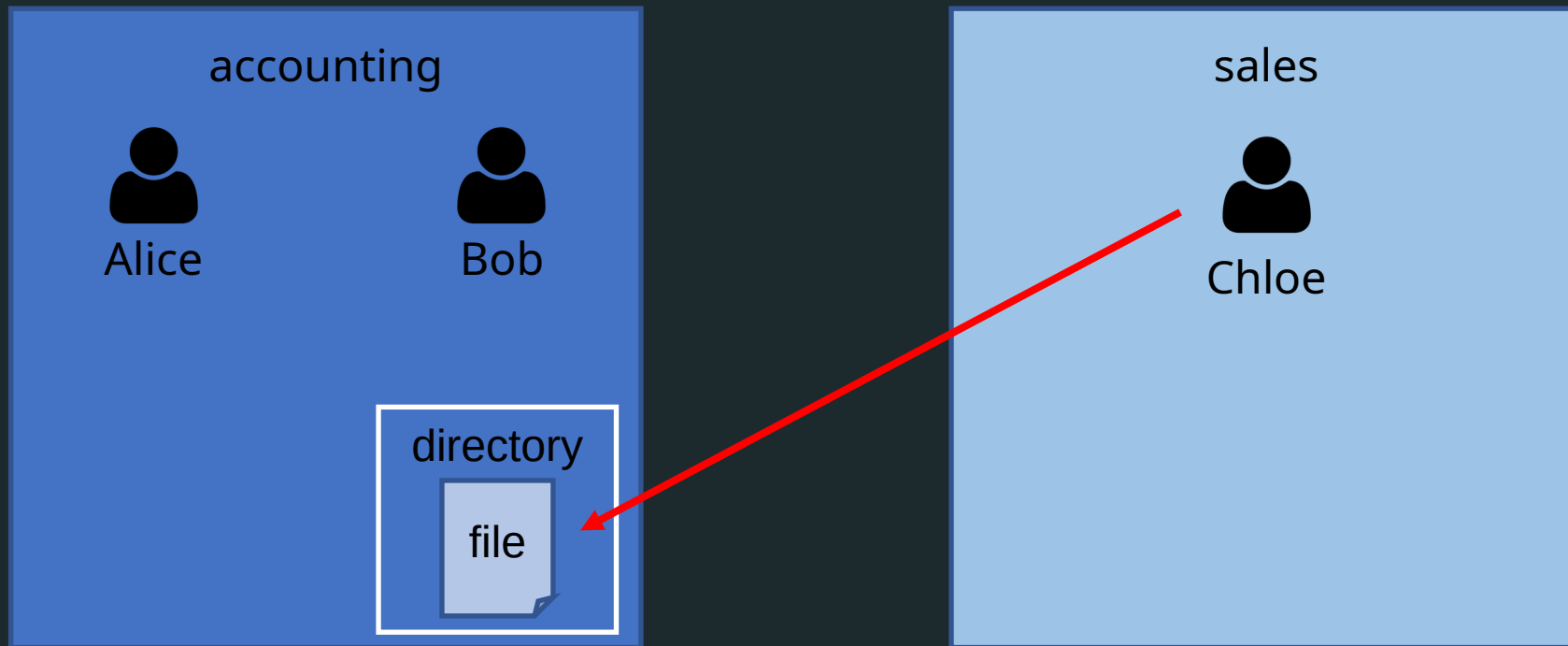
Linux ACL

```
└─(user1 @ kali) - [/home/dep1]
└─$ getfacl test
# file: test
# owner: user1
# group: dep1
user::rw-
user:dep1:rwx          #effective:rw-
group::rwx             #effective:rw-
mask::rw-
other::r--
```

Effective permissions are formed by AND-ing the actual permissions with the mask. Since the mask of the file is rw-, all the effective permissions have the x bit turned off.

Demo

Linux ACL



Introduction

Setuid, setgid, sticky bit

- On a Unix-like operating system, the ownership of files and directories is based on the default **uid** (user-id) and **gid** (group-id) of the user who created them.
- The same thing happens when a process is launched: **it runs with the effective user-id** and group-id of the user who started it, and with the corresponding privileges.
- This behavior can be **modified by using special permissions**.

setuid

Setuid, setgid, sticky bit

- When the **setuid** bit is used, the behavior described above it's modified so that when an executable is launched, **it does not run with the privileges of the user who launched it, but with that of the file owner instead.**
- So, for example, if an executable has the setuid bit set on it, and it's owned by root, when launched by a normal user, it will run with root privileges.
- It should be clear why this represents a **potential security risk, if not used correctly.**
- An example of an executable with the setuid permission set is **passwd**, the utility we can use to change our login password.

setuid

Setuid, setgid, sticky bit



```
kali@kali: /bin
File Actions Edit View Help
kali@kali:/bin$ ls -la passwd
-rwsr-xr-x 1 root root 66292 Feb 7 2020 passwd
```

- How to identify the setuid bit? As you surely have noticed looking at the output of the command above, the setuid bit is represented by an **s** in place of the **x** of the executable bit.
- The **s** implies that the executable bit is set, otherwise you would see a capital **S**. This happens when the setuid or setgid bits are set, but the executable bit is not, **showing the user an inconsistency**: the setuid and setgid bits **have no effect** if the executable bit is not set.
- The setuid bit has no effect on directories.

setgid

Setuid, setgid, sticky bit

- Unlike the setuid bit, the setgid bit has effect on **both files and directories**.
- In the first case, the file which has the setgid bit set, when executed, instead of **running with the privileges of the group** of the user who started it, runs with those of the group **which owns the file**.
- When used on a directory, instead, the setgid bit alters the standard behavior so that the group of the **files created inside said directory**, will not be that of the user who created them, but that **of the parent directory itself**.

Sticky bit

Setuid, setgid, sticky bit

```
drwx----- 3 root root 4096 Oct 20 05:57 root
drwxr-xr-x 33 root root 820 Oct 20 06:48 run
lrwxrwxrwx 1 root root 8 Jul 27 13:10 sbin -> usr/sbin
drwxr-xr-x 3 root root 4096 Jul 27 13:29 srv
dr-xr-xr-x 13 root root 0 Oct 20 05:56 sys
drwxrwxrwt 13 root root 4096 Oct 20 10:19 tmp
drwxr-xr-x 13 root root 4096 Jul 27 13:13 usr
drwxr-xr-x 12 root root 4096 Jul 27 13:12 var
```

- The sticky bit works in a different way: while it has no effect on files, when used on a directory, **all the files in said directory will be modifiable only by their owners**. A typical case in which it is used, involves the */tmp* directory. Typically this directory is writable by all users on the system, so to make impossible for one user to delete the files of another one.
- The sticky bit is identifiable by a **t** which is reported where normally the executable x bit is shown, in the "other" section.

Command

Setuid, setgid, sticky bit

- Just like normal permissions, the special bits can be assigned with the `chmod` command, using the numeric or the `rwX` format. In the former case the `setuid`, `setgid`, and sticky bits are represented respectively by a value of 4, 2 and 1. So for example if we want to set the `setgid` bit on a directory we would execute:

```
$ chmod 2775 test
```

- The other way we can set the special permissions bits is to use the `rwX` syntax:

```
$ chmod g+s test      #setgid
$ chmod u+s file      #setuid
$ chmod o+t test      #sticky bit
```

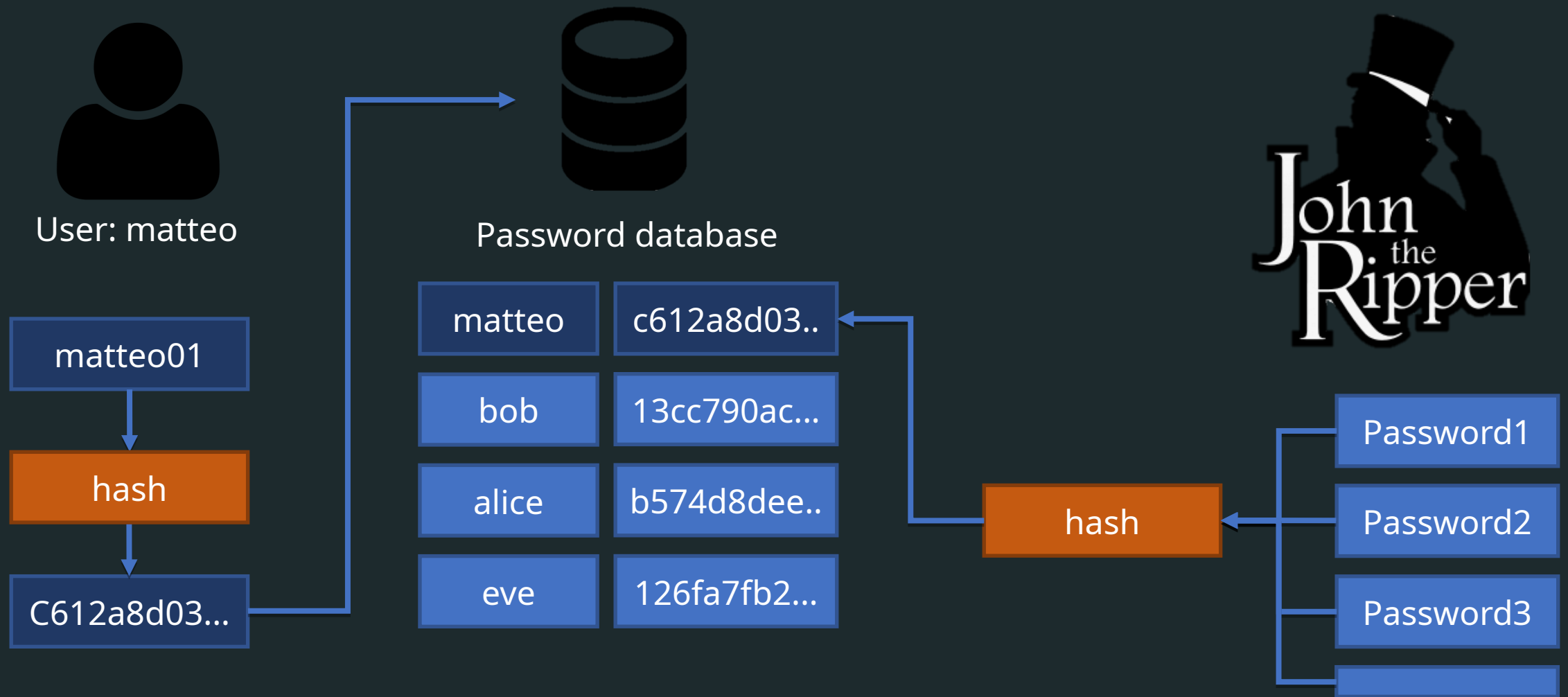
John the Ripper

Password cracking

- John the Ripper is a free password cracking software tool.
- One of the most frequently used password testing and breaking programs as it combines a number of password crackers into one package, auto detects password hash types, and includes a customizable cracker.
- It can be run against various encrypted password formats including several crypt password hash types most commonly found on various Linux distributions and Windows.

Password hash

Password cracking



Password file

Password cracking

- Traditional Unix systems keep user account information, including one-way encrypted passwords, in a text file called `/etc/passwd`. As this file is used by many tools (such as `ls`) to display file ownerships, etc. by matching user id #'s with the user's names, the file needs to be world-readable.
- `/etc/shadow` is a text file that contains information about the system's users' passwords as well as other information such as account or password expiration values etc. The `/etc/shadow` file is readable only by the root account (owned by user root and group shadow, 640 permissions) and is therefore less of a security risk.

Linux password file

Password cracking

File Actions Edit View Help

```
kali@kali:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
lightdm:x:131:139:Light Display Manager:/var/lib/lightdm:/bin/false
king-phisher:x:132:140::/var/lib/king-phisher:/usr/sbin/nologin
kali:x:1000:1000:kali,,,:/home/kali:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
accounting:x:1001:1001::/home/accounting:/bin/bash
Fred:x:1002:1001::/home/Fred:/bin/bash
sales:x:1003:1003::/home/sales:/bin/bash
Kenny:x:1004:1003::/home/Kenny:/bin/bash
Bill:x:1005:1001::/home/Bill:/bin/bash
kali@kali:~$
```

```
testuser:x:1481:1482:This is a test user:/home/testuser:/bin/bash
|      |      |      |
[Username] [Password] [Userid] [Groupid]
|
[User Information]
|
[User home path]
|
[User shell]
```

```
sudo unshadow /etc/passwd /etc/shadow > dump.txt
```

Unsalted password

Password cracking

```
Alice:4420d1918bbcf7686defdf9560bb5087d20076de5f77b7cb4c3b40bf46ec428b  
jason:695ddccd984217fe8d79858dc485b67d66489145afa78e8b27c1451b27cc7a2b  
mario:cd5cb49b8b62fb8dca38ff2503798eae71bfb87b0ce3210cf0acac43a3f2883c  
teresa:73fb51a0c9be7d988355706b18374e775b18707a8a03f7a61198eefc64b409e8  
bob:4420d1918bbcf7686defdf9560bb5087d20076de5f77b7cb4c3b40bf46ec428b  
mike:77b177de23f81d37b5b4495046b227befa4546db63cfe6fe541fc4c3cd216eb9
```

- A **rainbow table** can make the exploitation of unsalted passwords easier. A rainbow table is essentially a **pre-computed database of hashes**. Dictionaries and random strings are run through a selected hash function and the input/hash mapping is stored in a table.

Salted password





Password cracking

- The attacker can then simply do a password **reverse lookup** by using the hashes from a stolen password database.
- The main difference between a rainbow table attack and a dictionary and brute-force attack is pre-computation. Rainbow table attacks are fast because the attacker **doesn't have to spend any time computing any hashes**.
- The trade-off for the speed gained is the **immense amount of space** required to host a rainbow table

Salted password

Password cracking

- To mitigate the damage that a rainbow table or a dictionary attack could do, we salt the passwords. According to OWASP Guideliness, a salt is a fixed-length cryptographically-strong **random value that is added to the input** of hash functions to create unique hashes for every input.

				
Password	p4s5w3rdz	p4s5w3rdz	p4s5w3rdz	p4s5w3rdz
Salt	-	-	et52ed	ye5sf8
Hash	f4c31aa	f4c31aa	1vn49sa	z32i6t0

John cracking modes

Password cracking

- **Single:** it use the login names, "GECOS" / "Full Name" fields, and users' home directory names as candidate passwords with a large set of mangling rules applied.
- **Wordlist:** simplest cracking mode, John try to hash all word specified in a wordlist file (a text file containing one word per line) also called dictionary and check if the password hash matches.
- **Incremental:** this is the most powerful cracking mode, it can try all possible character combinations as passwords. However, it is assumed that cracking with this mode will never terminate because of the number of combinations being too large.
- **External**

Demo

Password cracking

demo

John utils

Password cracking

- <https://countuponsecurity.files.wordpress.com/2016/09/jtr-heat-sheet.pdf>
- <https://wiki.skullsecurity.org/Passwords>
- <https://www.openwall.com/john/doc/MODES.shtml>
- [https://cheatsheetseries.owasp.org/cheatsheets/Password Storage Cheat Sheet.html#Use a cryptographically strong credential-specific salt](https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html#Use_a_cryptographically_strong_credential-specific_salt)