Sicurezza dei sistemi e permessi, Linux

Fondamenti di Cybersecurity 2022/2023

Davide Berardi <davide.berardi@unibo.it>

La sicurezza dei sistemi differisce da quella delle reti.

- Possibilità di avere un ente certificato nel mezzo (Sistema operativo);
- I sistemi possono essere singolo o multi-utente;
- Possibiltà di divisione dei privilegi.

È quella su cui agiscono Malware locali.

Si definisce privilege escalation la possibilità di ottenere più privilegi sul sistema.

Ad esempio la possiblità di installare programmi, leggere informazioni private o modificare configurazioni del sistema.

Il primo passo per il privilege escalation è l'esecuzione arbitraria di codice (p.e. per poter effettuare attacchi per ottenere privilegi più elevati).

Questo può essere perpetrato:

- Installando software.
- Facendo girare software.
- Sfruttando vulnerabilità in grado di dirottare il funzionamento dei programmi.

In UNIX (più o meno!) tutto è un file.

Questo significa che avendo controllo su un file si può ottenere controllo su quello che rappresenta (e.g. /dev/snd/* per l'audio).

Il sistema mantiene le informazioni di accesso ai file tramite un meccanismo denominato Access Control List (ACL). Queste vengono rappresentate come una tabella soggetto / oggetto:

Esempio: Alice: read,write ; Bob: read ; Other: read

Unix / Linux (senza le estensioni denominate Posix ACL), dichiara 3 soggetti:

- Il proprietario di un file
- Il gruppo proprietario di un file
- Tutti gli altri

```
bera@snark ~ % ls -la /etc/passwd
-rw-r--r-- 1 root root 2083 Mar  9 17:23 /etc/passwd
```

I soggetti sono identificati tramite un file (/etc/passwd), il quale contiene un'associazione nome utente: user id.

Lo user id viene quindi salvato nel file-system, associato ad ogni file.

```
tor:x:43:43::/var/lib/tor:/usr/bin/nologin
usbmux:x:140:140:usbmux user:/:/usr/bin/nologin
```

Analogamente, per i gruppi, esiste un file (/etc/group) e un group id per gruppo.

```
vboxusers:x:108:bera
dhcpcd:x:986:
```

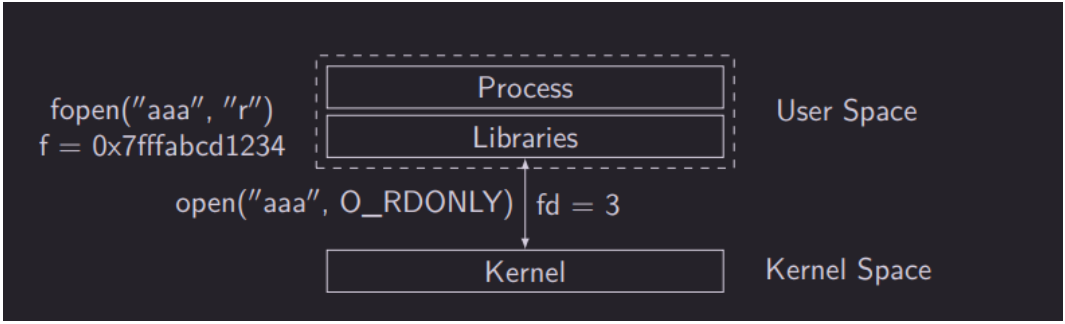
Normalmente in UNIX (sistemi multi-utente) è possibile **regalare** l'accesso a file ad altri utenti.

(vedremo dopo il comando chmod)

Una capability è un token in grado di rappresentare un determinato oggetto sul quale si ha l'accesso. Alcuni esempi di capability sono:

- I cookie web (identificatori che indicano a quale sessione è associata una comunicazione).
- I file aperti.

A seguito di una system call (richiesta al sistema operativo) open, viene ritornato un identificativo univoco che può essere riconosciuto dal sistema operativo per indicare il file.


fopen("aaa", "r")
f = 0x7fffabcd1234
Process
Libraries
User Space
open("aaa", O_RDONLY)
fd = 3
Kernel
Kernel Space

Le capability possono essere supportate da sistemi crittografici (e.g. cookie) o segmentazione a livello di sistema operativo (e.g. file).

Unix utilizza un sistema misto tra ACL (file non aperti) e Capabilities (file aperti / socket / ...).

Se volessimo evitare il regalo di accesso da parte dell'utente owner è possibile utilizzare un sistema di Mandatory Access Control (MAC).

In Linux ne esistono diversi:

- SeLinux
- Tomoyo
- SMACK
- ...

È facile pensare a questi sistemi pensando alla segregazione delle Applicazioni Android. E.g. da Telegram non posso vedere i messaggi di Whatsapp.

Esistono diversi modelli di flusso delle informazioni. Supponiamo di avere diversi file a diversi livelli di segretezza. Il modello Bell-LaPadula mette in atto le seguenti regole:

- Ogni soggetto e ogni oggetto hanno un livello di sicurezza.
- Un soggetto può leggere oggetti a livelli di sicurezza minori o uguali al suo.
- Un soggetto può scrivere oggetti a livelli di sicurezza pari o maggiori del suo.

WURD: Write-Up-Read-Down

Supponiamo di avere tre livelli di sicurezza:

- Top-Secret
- Secret
- Public

Alice è un'operatrice a livello Secret. Alice può:

- Leggere documenti Secret e Public
- Scrivere documenti Secret e Top-Secret

Alice non può scrivere documenti pubblici o leggere documenti top-secret!

Biba è un modello duale a Bell-LaPadula.

- Ogni soggetto e ogni oggetto hanno un livello di sicurezza.
- Un soggetto può scrivere oggetti a livelli di sicurezza minori o uguali al suo.
- Un soggetto può leggere oggetti a livelli di sicurezza pari o maggiori del suo.

WDRU: Write-Down-Read-Up

Nel mondo Windows prende il nome di Integrity Level (IL)

Usato per l'integrità dei dati. Supponiamo di avere tre livelli di sicurezza:

- Capitano
- Tenente
- Carabiniere Scelto

Alice è un'operatrice a livello Tenente. Alice può:

- Leggere documenti rilasciati da Tenenti e Capitani
- Scrivere documenti per Tenenti e Carabinieri Scelti

In questo modo Alice non può dare ordini a un suo superiore!

Nei sistemi operativi, il sistema per lo scambio dei dati locali (su cui poi si basa la logica UNIX) è il filesystem.

Su Linux a esempio si compone di una radice (/) dalla quale il sistema si dirama come un albero.

`/`	The root filesystem
`/proc`	(pseudo) The process virtual infrastructure (e.g. /proc/1/cmdline).
`/sys`	(pseudo) The system virtual infrastructure (e.g. /sys/class/leds).
`/dev`	(pseudo) Device drivers file interface (e.g. /dev/sda).
`/run`	(pseudo) Contains run-time information.
`/etc`	Configuration directory.
`/tmp`	Temporary directory, volatile, usually mounted in RAM.
`/root`	Root's home.
`/bin`	Binaries.
`/lib`	Libraries
`/sbin`	System Binaries.
`/usr`	User binaries
`/var`	Log, cache, and structured personal files (e.g. mailboxes).
`/home`	Users' directories.
`/mnt`	External mountpoint.
`/opt`	Optionals softwares.

Come dichiarato precedentemente, il sistema in modalità DAC (senza SeLinux o meccanismi MAC) analizza (in questo ordine!!!): UID processo e proprietario del file; GID e gruppo del file ; UID e Other.

Può generare complicanze contro-intuitive!!! (esempio se l'utente appartiene a un gruppo che può leggere il file ma è il suo proprietario e il proprietario non può leggere il file)

Il super utente privilegiato del sistema è quello che può svolgere ogni operazione a partire dalla radice (root) del file system.

Questo utente prende il nome di root, SYSTEM nei sistemi Windows.

È possibile cambiare il proprietario (o il gruppo) di appartenenza di un file.

Solo root (normalmente) può regalare i file agli altri!!!

È possibile anche modificare i permessi associati a un file. Questi vengono configurati tramite un comando chiamato chmod.

Questo comando prevede un'interfaccia a linea di comando in grado di interpretare numeri in base 8 (ottali) o un sistema di configurazione più "intuitivo" basato su un linguaggio specifico:

- bit 0: execute, possibilità di eseguire il file.
- bit 1: write, possibilità di scrivere il file
- bit 2: read, possibilità di leggere il file

Ovviamente impostando 777 come permessi ottali a un file lo stiamo regalando a tutti (other: read, write, execute)!!!

Sulle cartelle i permessi hanno un significato lievemente diverso:

- bit 0: execute, possibilità di entrare in una cartella.
- bit 1: write, possibilità di eliminare i file contenuti in una cartella (anche quelli non nostri su cui non abbiamo permesso di scrittura!!!!)
- bit 2: read, possibilità di leggere il contenuto di una cartella.

C'è un caso di security by obscurity, quale? :)

C'è un caso di security by obscurity, quale? :)

chmod 711 test/

Un utente può appartenere a più gruppi, questi vengono usati di solito per controllare accessi a file di device driver (/dev).

```
bera@snark ~ % ls -la /dev/tty0
crw--w---- 1 root tty 4, 0 Mar 23 16:25 /dev/tty0
```

Esistono 3 permessi speciali, i quali sono salvati nei bit più significativi dei metadati del file:

- setuid
- setgid
- sticky

Una cartella con permesso setuid verrà ignorata, mentre setgid assegnerà ai file creati al suo interno il gruppo d'appartenenza della cartella.

Questo è utile per mantenere all'interno di una cartella il gruppo corretto per tutti i file.

Lo sticky bit su una cartella invece permetterà l'eliminazione dei file al suo interno solo al suo owner (anche se la cartella ha permesso di scrittura per tutti, e.g. /tmp/).

Lo sticky bit su una file è ignorato e deprecato.

Un file setuid verrà eseguito come se fosse eseguito dal suo proprietario.

Analogamente un file setgid eseguirà con il gruppo di appartenenza.

ESTREMAMENTE PERICOLOSO!!!

Cosa succede, ad esempio se usiamo il seguente codice C?

```
...
system("whoami");
...
```

Il sistema eseguirà il comando whoami con i privilegi dell'utente indicato.

whoami è un comando shell, che può quindi essere dirottato cambiandogli il "nome".

PATH=. ./vulnprogram

In questo modo funziona il comando sudo, esegue con privilegi elevati e controlla la password dell'utente (normalmente tramite un framework chiamato PAM).

Description

In Sudo before 1.9.12p2, the sudoedit (aka -e) feature mishandles extra arguments passed in the user-provided environment variables (SUDO_EDITOR, VISUAL, and EDITOR), allowing a local attacker to append arbitrary entries to the list of files to process. This can lead to privilege escalation. Affected versions are 1.8.0 through 1.9.12.p1. The problem exists because a user-specified editor may contain a "--" argument that defeats a protection mechanism, e.g., an EDITOR='vim -- /path/to/extra/file' value.

Severity

CVSS Version 3.x | CVSS Version 2.0

CVSS 3.x Severity and Metrics:

NIST: NVD **Base Score:** 7.8 HIGH **Vector:** CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H

setuid è un ovvio problema di sicurezza. Per ottenere uno dei privilegi di root (e.g. cambiare un file) otteniamo l'intero insieme dei suoi privilegi (e.g. la possibilità di cambiare indirizzo IP della macchina). Per questo motivo sono stati spezzettati i privilegi di root in vari sotto privilegi, ad esempio:

- CAP_NET_ADMIN
- CAP_DAC_OVERRIDE
- ... (man capabilities)

Domande?