Programmazione Sicura

Esecuzione con privilegi elevati

Barbara Masucci

UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA

DIPARTIMENTO DI ECCELLENZA

Punto della situazione

- Nelle lezioni scorse abbiamo visto alcuni sistemi per catalogare vulnerabilità e debolezze

- Scopo della lezione di oggi:
  - Introdurre il problema dell'esecuzione con privilegi elevati nei sistemi Unix-like
  - Descrivere la politica di controllo degli accessi nei sistemi Unix-like

Esecuzione con privilegi elevati

- Nei sistemi UNIX può capitare che un processo necessiti di privilegi elevati (tipicamente, privilegi di `root`)

- Alcuni esempi
  - Il comando `passwd` necessita dei permessi di `root` per modificare il file `/etc/passwd`
  - Il comando `ping` necessita dei permessi di `root` per aprire socket e porte
  - Il comando `mount` necessita dei permessi di `root` per montare file system residenti su memorie di massa removibili

Esecuzione con privilegi elevati

L'elevazione dei privilegi può essere effettuata in due modi

- Manuale
  1. L'utente digita i comandi opportuni per diventare un amministratore
  2. L'utente esegue il comando che gli interessa
- Automatica (tipicamente usata in UNIX)
  1. L'utente esegue il comando che gli interessa
  2. Il SO esegue l'elevazione dei privilegi

Esempio di elevazione manuale

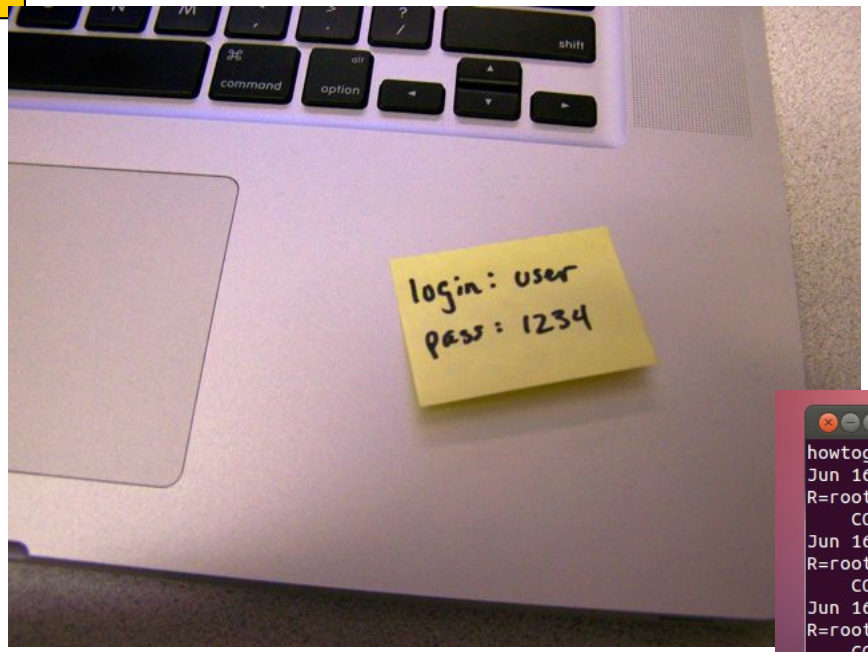
```
$ su –
Password:
$ passwd masucci
Changing password for user masucci.
New password:
Retype new password:
passwd: all authentication tokens updated succesfully
```

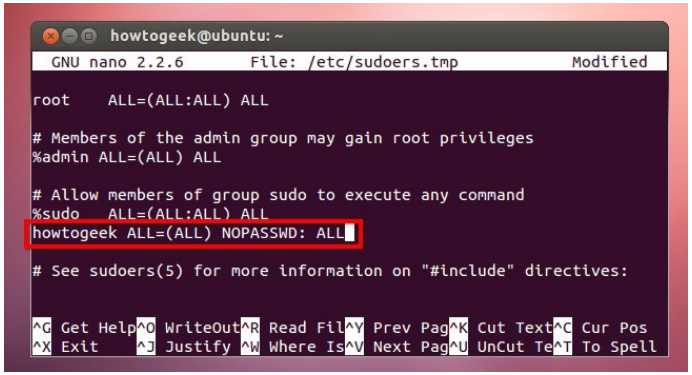
- Il comando `su` consente a un utente eseguire comandi con i privilegi di un altro utente, in questo caso l'utente `root`
- La procedura richiede la conoscenza della password di `root`
- Una volta acquisiti i privilegi di `root`, l'utente può modificare la sua password (che sarà scritta nel file `/etc/passwd`)

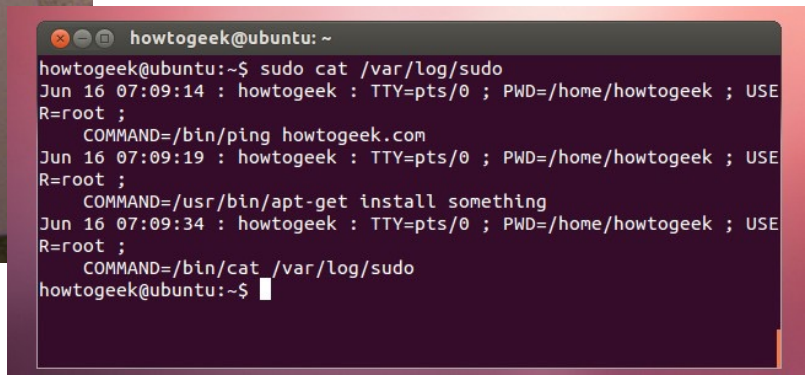
Elevazione manuale

- Con questo sistema è necessario conoscere la password di `root`
  - Di conseguenza, l'utente può ispezionare il sistema, modificare file e compromettere servizi
- Inoltre è necessario immettere la password di `root` ogni volta che serve acquisire privilegi elevati
  - C'è il rischio che per agevolare l'operazione …si decida di non impostare alcuna password per la `root`!

Conseguenze







DISASTRO:
Chiunque accede al terminale
può diventare amministratore!

Esecuzione con privilegi elevati

L'elevazione dei privilegi può essere effettuata in due modi

- Manuale
  1. L'utente digita i comandi opportuni per diventare un amministratore
  2. L'utente esegue il comando che gli interessa
- Automatica (tipicamente usata in UNIX)
  1. L'utente esegue il comando che gli interessa
  2. Il SO esegue l'elevazione dei privilegi

Esempio di elevazione automatica

```
$ passwd masucci
Changing password for user masucci.
Current password:
New password:
Retype new password:
passwd: all authentication tokens updated succesfully
```

- L'utente richiede di modificare la password mediante il comando `passwd`
- Il SO effettua l'elevazione automatica ai privilegi di `root`, consentendo all'utente di modificare la password (che sarà scritta nel file `/etc/passwd`)

Elevazione automatica

- Come è possibile che l'utente abbia ottenuto i privilegi di `root` per l'esecuzione del comando `passwd`?
- Per comprenderlo, rivediamo brevemente l'organizzazione del file system nei sistemi UNIX

Utenti e gruppi

- In UNIX, un utente è caratterizzato da
  - Username (stringa)
  - UID, User IDentification number (intero)
  - La root è l'utente con UID=0
- Un gruppo è caratterizzato da
  - Groupname (stringa)
  - GID, Group IDentification number (intero)
- Queste informazioni possono essere visualizzate con il comando `id`

File e permessi

- L'accesso ai file è regolato da permessi, definiti come tre terne di azioni (una terna per ciascuna tipologia di utenti)
  - Azioni: Read, Write, eXecute
  - Tipologie di utenti: proprietario, gruppo di lavoro, altri utenti

- Nota:
  - eXecute su un file indica che il file può essere eseguito
  - eXecute su una directory indica che si può entrare in essa

File e permessi

Rappresentazione dei permessi

- Rappresentazione ottale
  - Read → 4, Write → 2, eXecute → 1
  - Una terna di azioni → una somma di permessi
  - rwxrwxr-x → 4+2+1 4+2+1 4+1 → 775

- Rappresentazione simbolica
  - Read → r, Write → w, eXecute → x
  - Creatore → u, Gruppo → g, Altri → o
  - Una modifica → utenti ± azioni (u+rw)
  - Permesso finale → modifiche separate da ,
  - rwxrwxr-x → ug+rwx, o+rx

Impostare i permessi

- I permessi possono essere impostati con il comando

```
chmod nuovi-permessi nome-file
```

#	Permission	rwx
7	read, write and execute	rwx
6	read and write	rw-
5	read and execute	r-x
4	read only	r--
3	write and execute	-wx
2	write only	-w-
1	execute only	--x
0	none	---

File e permessi

- Dalla discussione precedente si evince che a ciascun file sia associata una stringa di permessi costituita da 9 bit
- In realtà ci sono tre bit aggiuntivi associati ai permessi: SETUID, SETGID, STICKY
- Di seguito discutiamo in dettaglio l'utilizzo dei primi due, tralasciando il terzo

SETUID e SETGID

- I bit aggiuntivi SETUID e SETGID sono molto rilevanti per la nostra discussione
- Quando sono posti uguali a 1, consentono una elevazione dei privilegi, come vedremo di seguito
- La rappresentazioni ottale / simbolica è
  - Per SETUID: 4 / s
  - Per SETGID: 2 / s

Impostare SETUID

- Per impostare a 1 il bit SETUID di un file, si utilizza il comando

```
chmod u+s file
```

SETUID

- Come stabilire se questi il bit SETUID è posto a 1 per un determinato file?
  - Invochiamo `ls –l nomefile` e vediamo se è presente il permesso "s" e se il file è evidenziato in rosso

```
barbara@barbara-VirtualBox:/usr/bin$ ls -la passwd
-rwsr-xr-x 1 root root 54256 mar 26  2019 passwd
```

Eseguibili con SETUID acceso

- Per individuare **tutti** i file con il bit SETUID acceso possiamo usare utilizzare il comando `find` con l'opzione `–perm` (filtro in base ai permessi)

```
find / -perm /u+s
```

- Per evitare di visualizzare i messaggi di errore (permission denied)

```
find / -perm /u+s 2>/dev/null
```

Esercizio

- Individuare i file con il bit SETUID acceso nella cartella `/usr/bin`

```
barbara@barbara-VirtualBox:~$ find /usr/bin -perm /u+s
/usr/bin/chfn
/usr/bin/passwd
/usr/bin/gpasswd
/usr/bin/chsh
/usr/bin/sudo
/usr/bin/newgrp
/usr/bin/pkexec
```

- Controllare permessi di ciascun file

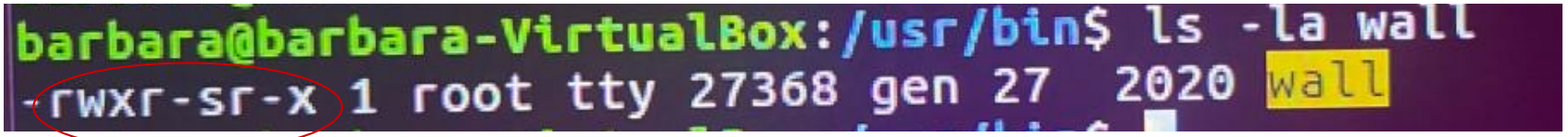
Impostare SETGID

- Per impostare a 1 il bit SETGID di un file, si utilizza il comando

```
chmod g+s file
```

SETGID

- Come stabilire se questi il bit SETGID è posto a 1 per un determinato file?
  - Invochiamo `ls –l nomefile` e vediamo se è presente il permesso "s" e se il file è evidenziato in giallo



Eseguibili con SETGID acceso

- Per individuare **tutti** i file con il bit SETGID acceso possiamo usare utilizzare il comando `find` con l'opzione `–perm` (filtro in base ai permessi)

```
find / -perm /g+s
```

- Per evitare di visualizzare i messaggi di errore (permission denied)

```
find / -perm /g+s 2>/dev/null
```

Utente e gruppo reale

- Nei SO UNIX, il descrittore di un processo memorizza una prima coppia di credenziali
  - Real User ID (RUID) è l'UID di chi ha lanciato il comando
  - Real Group ID (RGID) è il GID di chi ha lanciato il comando

Utente e gruppo effettivo

- Nei SO UNIX, il descrittore di un processo memorizza una seconda coppia di credenziali
  - Effective User ID (EUID)
  - Effective Group ID (EGID)

- In condizioni normali (bit SETUID e SETGID disattivati)
  - EUID è l'UID di chi ha lanciato il comando
  - EGID è il GID di chi ha lanciato il comando

Utente e gruppo effettivo

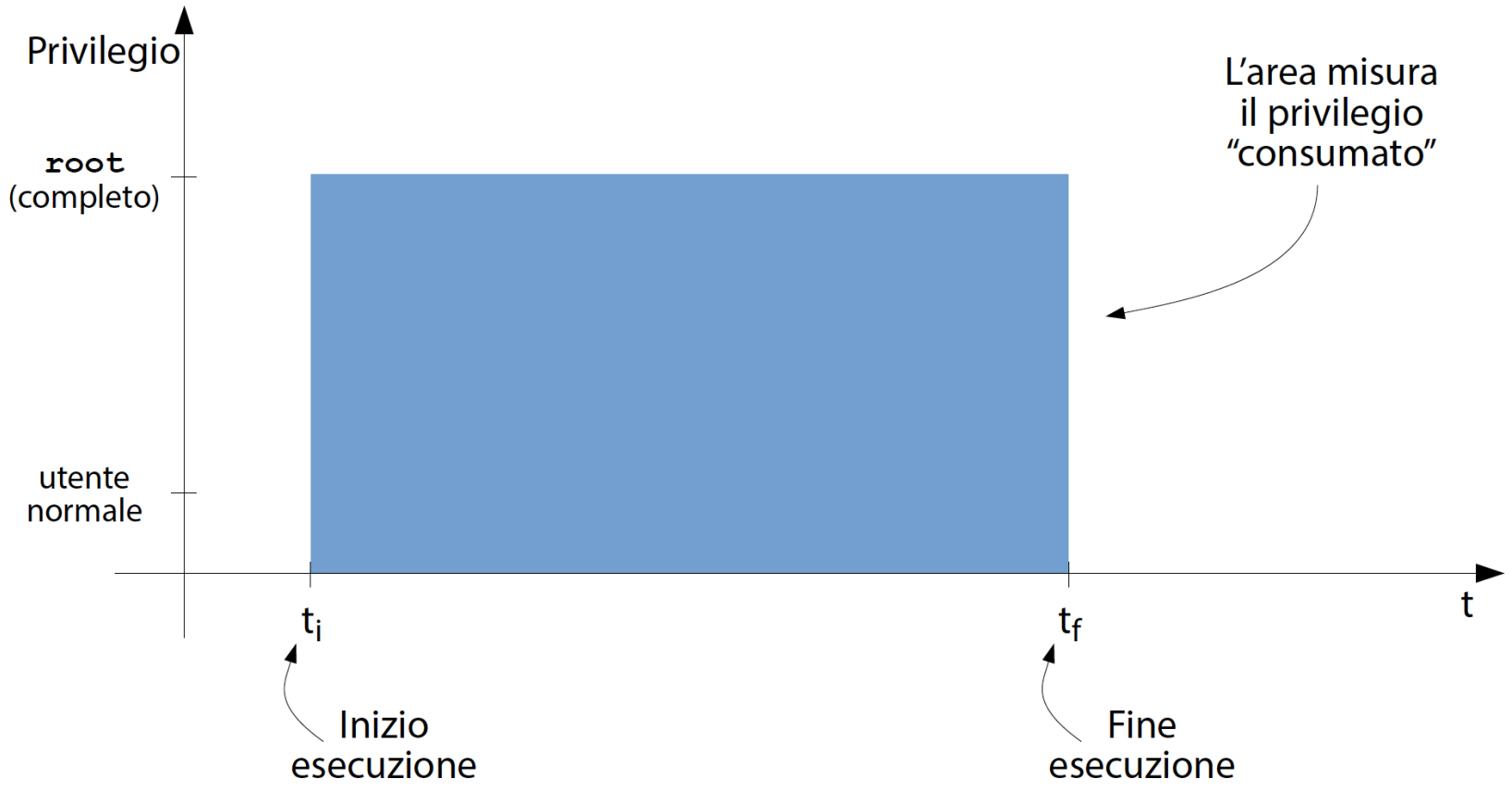
- Se invece i bit SETUID e SETGID sono attivati
  - EUID è l'UID di chi possiede il file
  - EGID è il GID di chi possiede il file

Quindi, chi lancia il comando può assumere i privilegi di chi possiede il file!

Problemi

- Se non sono presenti altre contromisure, un processo con SETUID/SETGID attivi ottiene i pieni poteri dell'utente privilegiato
  - Se l'utente privilegiato è `root`, il processo può fare di tutto
  - Inoltre, il privilegio ottenuto è mantenuto per l'intera esecuzione

Un grafico esplicativo



Debolezze? SI

- L'elevazione automatica dei privilegi è una funzionalità interessante offerta da UNIX

- Tuttavia
  - Il conferimento dei pieni poteri di `root`, quando non sono strettamente necessari, è una debolezza
  - Anche il mantenimento dei privilegi per tutta la durata dell'esecuzione è una debolezza

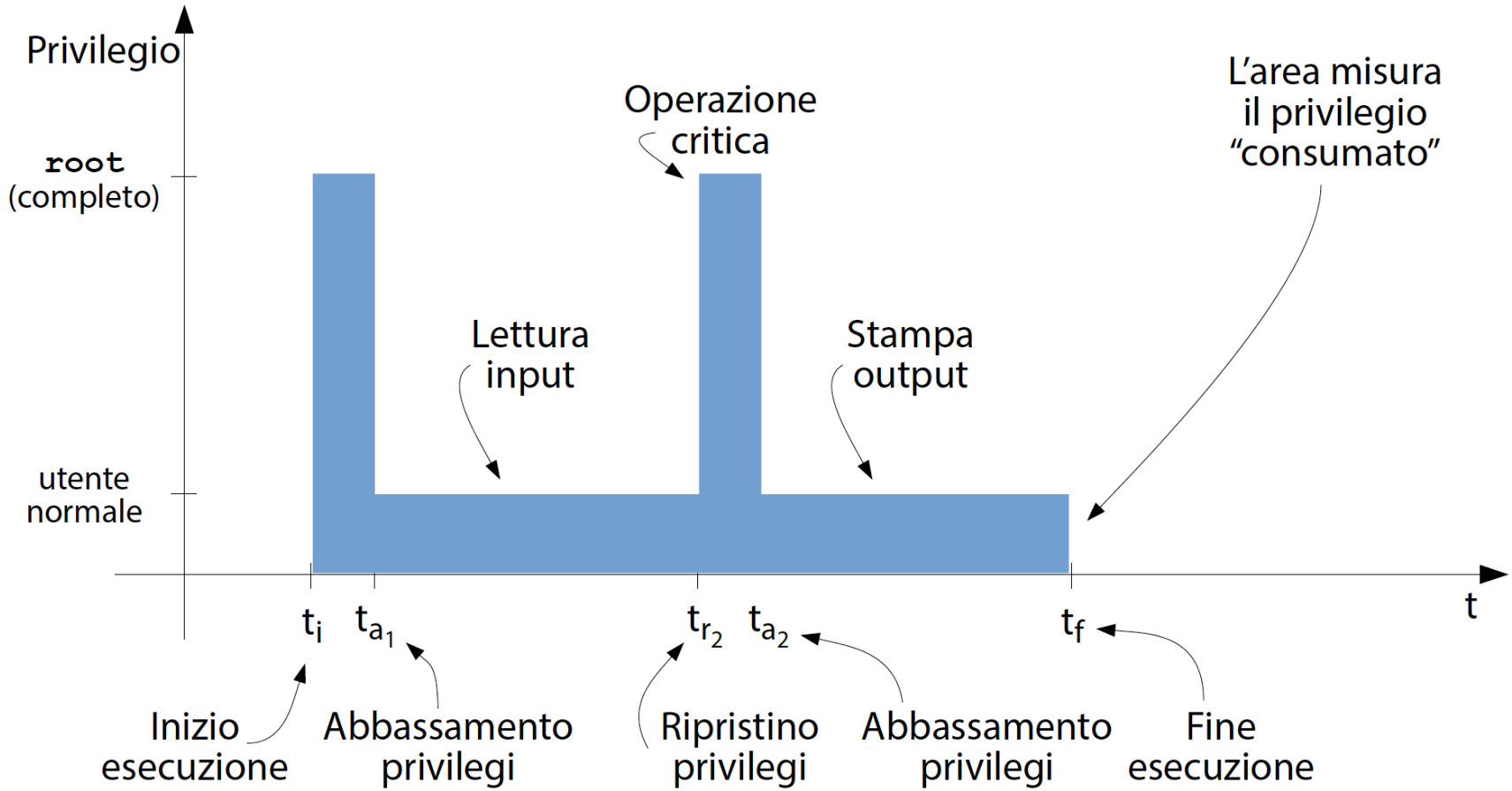
Vulnerabilità? NO

- Non si tratta di vulnerabilità, perchè le debolezza non sono sfruttabili da sole!
  - Se un programma è scritto in modo corretto, la sola elevazione automatica non può dare alcun vantaggio a un attaccante
  - Tuttavia, se unita ad altre debolezze, l'elevazione automatica dei privilegi può avere conseguenze devastanti per la sicurezza di un programma

Come risolvere?

- Abbassamento e ripristino dei privilegi
  - Se l'applicazione non svolge operazioni critiche, può decidere di abbassare i propri privilegi a quelli dell'utente che ha eseguito il comando (privilege drop)
  - Quando l'applicazione svolge operazioni critiche, ripristina nuovamente i privilegi ottenuti con l'elevazione automatica (privilege restore)

Un grafico esplicativo



Privilege drop and restore

- Come si implementano l'abbassamento ed il ripristino dei privilegi di un programma?
- Vedremo quali sono le chiamate di sistema che assolvono al compito, relativamente a diversi sistemi
  - I primissimi sistemi UNIX
  - I sistemi UNIX System V
  - I sistemi UNIX BSD
  - I sistemi POSIX
  - I sistemi UNIX moderni

Primissimi sistemi UNIX

- Nei primissimi sistemi UNIX sono previste solo due tipologie di user (e group) ID:
  - RUID (RGID)
  - EUID (EGID)

- I valori RUID ed EUID possono essere determinati mediante le chiamate di sistema
  - `getuid()`, che restituisce il RUID del processo invocante
  - `geteuid()`, che restituisce l'EUID del processo invocante

Primissimi sistemi UNIX

- E' anche possibile cambiare il valore dell'EUID di un processo
- La chiamata di sistema `setuid(uid)` imposta l'EUID del processo al valore `uid` in input
  - Ad esempio, `setuid(0)` imposta EUID a `0` (`root`)
  - In caso di errore, l'output della chiamata è -1
- Per effettuare un abbassamento dei privilegi basta eseguire il comando `setuid(getuid());`
  - In tal modo infatti, l'EUID del processo viene posto uguale al RUID

Primissimi sistemi UNIX

- E' possibile ripristinare i privilegi elevati dopo averli abbassati?
  - No. Nei primissimi sistemi UNIX, l'abbassamento dei privilegi tramite lo statement

    ```
    setuid(getuid());
    ```

    non permette più il ripristino del privilegio elevato che si aveva in precedenza
  - Si tratta quindi di un abbassamento permanente dei privilegi

Primissimi sistemi UNIX

- Supponiamo che un processo
  - Parta SETUID `root`
  - Abbassi i suoi privilegi con `setuid(getuid())`
  - Invochi `setuid(0)` per ripristinare i privilegi di `root`
  - Viene generato un errore
- La possibilità di un abbassamento temporaneo dei privilegi viene gestita nei sistemi UNIX System V
  - Viene introdotta una nuova chiamata di sistema: `seteuid(uid)`

I sistemi UNIX System V

- Analogamente alla vecchia `setuid(uid)` anche la nuova chiamata di sistema `seteuid(uid)` imposta l'EUID del processo al valore `uid`

- Per effettuare un abbassamento dei privilegi basta eseguire il comando

```
seteuid(getuid());
```

I sistemi UNIX System V

- Supponiamo che un processo
  - Parta SETUID `root`
  - Abbassi i suoi privilegi con `seteuid(getuid())`
  - Invochi `seteuid(0)`

- Allora `seteuid(0)`
  - Termina correttamente, perchè `root` è l'utente effettivo
  - Imposta l'EUID al valore 0 (`root`)

E' un ripristino di privilegi!

I sistemi UNIX System V

- Abbiamo quindi due differenti modalità di abbassamento dei privilegi
  - Permanente
    ```
    setuid(getuid());
    ```
  - Temporaneo
    ```
    seteuid(getuid());
    ```

I sistemi UNIX BSD

- Nei sistemi BSD viene introdotta una chiamata di sistema per impostare contemporaneamente RUID e EUID:

  `setreuid(uid,euid)`

- Per lasciare inalterato uno dei due valori, basta fornire in input -1

I sistemi UNIX BSD

- L'abbassamento permanente dei privilegi si ottiene impostando entrambi i parametri ad un valore non privilegiato

```
setreuid(uid,uid);
```

- In seguito, i parametri RUID e EUID non potranno che assumere il valore `uid`
  - Il ripristino a `root` è impossibile

I sistemi UNIX BSD

- L'abbassamento temporaneo dei privilegi si ottiene con la chiamata
  `setreuid(geteuid(),getuid());`
- Nota: RUID ed EUID sono scambiati
- Dopo lo scambio:
  - RUID è da utente privilegiato
  - EUID è da utente non privilegiato

I sistemi UNIX BSD

- Il ripristino dei privilegi si ottiene ancora con la chiamata

```
setreuid(geteuid(),getuid());
```

- Dopo lo scambio:
  - RUID è da utente non privilegiato
  - EUID è da utente privilegiato

I sistemi POSIX

- POSIX (Portable Operating System Interface) è una famiglia di standard specificati dalla IEEE Computer Society per mantenere la compatibilità tra diversi SO

- Rilasciato nel 1988 con il nome IEEE 103 o ISO/IEC 9945, lo standard definisce
  - API di sistema
  - Interfaccia da linea di comando
  - Applicazioni di base

I sistemi POSIX

- Nei sistemi POSIX la chiamata di sistema `setreuid(uid,uid)` è stata deprecata e viene sostituita da `setuid(uid)`

- Anche nei sistemi POSIX abbiamo due differenti modalità di abbassamento dei privilegi
  - Permanente
    `setuid(getuid());`
  - Temporaneo
    `seteuid(getuid());`

I sistemi UNIX moderni

- I SO GNU/Linux implementano la LSB (Linux Standards Base)
  - Si tratta di una evoluzione della SUS (Single UNIX Specification), che è l'evoluzione attuale dello standard POSIX
- LSB sostituisce alcune funzionalità errate od insicure di POSIX/SUS con meccanismi propri

I sistemi UNIX moderni

- A partire dai SO UNIX System V è presente una terza coppia di credenziali:
  - Saved User ID (SUID)
  - Saved Group ID (SGID)
  - Quando un processo parte SUID=EUID, SGID=EGID
- Non vengono però fornite chiamate di sistema per gestire SUID e SGID
- Tale limite viene superato nei sistemi UNIX moderni

I sistemi UNIX moderni

- La chiamata di sistema `getresuid(uid,euid,suid)` consente di recuperare tutti gli user ID del processo invocante
- La chiamata di sistema `setresuid(uid,euid,suid)` consente di impostare tutti gli user ID del processo invocante

I sistemi UNIX moderni

- L'abbassamento permanente dei privilegi si ottiene impostando tutti i parametri ad un valore non privilegiato

```
setresuid(uid,uid,uid);
```

- In seguito, i parametri RUID, EUID e SUID non potranno che assumere il valore `uid`
  - Il ripristino a `root` è impossibile

I sistemi UNIX moderni

- L'abbassamento temporaneo dei privilegi si ottiene impostando EUID ad un valore non privilegiato
  `setresuid(-1,geteuid(),-1);`
- In questo modo si preserva lo user ID salvato e si può effettuare il ripristino in seguito
- Nota: `setresuid(-1,uid,-1)=seteuid(uid)`

I sistemi UNIX moderni

- Il ripristino temporaneo dei privilegi si ottiene impostando EUID ad un valore privilegiato

```
setresuid(-1,privileged_ID,-1);
```

dove `privileged_id` è lo user ID dell'utente privilegiato ottenuto in partenza (tipicamente `root`)

Elevazione dei privilegi via SETGID

- L'API di gestione dei privilegi tramite SETGID è concettualmente identica a quella tramite SETUID
  - `getgid()` restituisce l'RGID del processo
  - `getegid()` restituisce l'EGID del processo
  - `setgid(gid)` imposta l'EGID del processo
  - `setegid(egid)` imposta l'EGID del processo
  - `getresgid(gid,egid,sgid)` restituisce RGID, EGID e SGID del processo
  - `setresgid(gid,egid,sgid)` imposta RGID, EGID e SGID del processo

Inibire SETUID e SETGID

- Il meccanismo dei privilegi basati su SETUID e SETGID fornisce i pieni poteri di `root`
  - Per tale motivo si preferisce inibirlo il più possible
  - L'obiettivo è quello di impedire l'esecuzione di shell con privilegi di `root`
- L'opzione `nosuid` inibisce l'elevazione dei privilegi tramite SETUID e SETGID
  - Tutti i file system virtuali e temporanei sono montati con tale opzione

Inibire SETUID e SETGID

- Di default, molte shell moderne inibiscono l'elevazione dei privilegi tramite SETUID e SETGID
- In tal modo si evitano attacchi in grado di provocare esecuzione di codice arbitrario
- Ad esempio, quando BASH esegue uno script con privilegi elevati, ne abbassa i privilegi a quelli dell'utente che ha invocato la shell

Inibire SETUID e SETGID

- Ad esempio, si copi `/bin/bash` in una directory non montata `suid`

```
cp /bin/bash $HOME
```

- La si renda SETUID `root`

```
sudo chown root ./bash
sudo chmod u+s ./bash
```

- La si esegua e si stampino gli ID reale ed effettivo

```
$HOME/bash
ps –p $$ -o ruid,rgid,euid,egid
```

Inibire SETUID e SETGID

- L'esecuzione è identica a quella in assenza di SETUID `root`
  - `bash` stampa RUID, RGID, EUID, EGID dell'utente (normale) che lo ha lanciato
  - Non vi è traccia di ID privilegiati
  - Ciò implica l'abbassamento dei privilegi da parte di `bash`

Inibire SETUID e SETGID

- Ad esempio, si crei uno script `scr.sh` con il seguente contenuto

```
#!/bin/bash
        id –u ← stampa EUID
        id –g ← stampa EGID
```

- Lo si renda SETUID `root`

```
sudo chown root ./scr.sh
sudo chmod u+s ./scr.sh
```

- Lo si esegua

```
./scr.sh
```

Inibire SETUID e SETGID

- L'esecuzione è identica a quella in assenza di SETUID `root`
  - Lo script `scr.sh` stampa EUID, EGID dell'utente (normale) che lo ha lanciato
  - Non vi è traccia di ID privilegiati
  - Ciò implica ancora l'abbassamento dei privilegi da parte di `bash`

Inibire SETUID e SETGID

- E' possibile inibire l'abbassamento dei privilegi con l'opzione `–p` di BASH

- Proviamo

```
./bash -p
ps –p $$ -o ruid,rgid,euid,egid
```

- BASH esegue con i privilegi di `root`!
  - Infatti mentre prima EUID era 1000, ora è 0

Inibire SETUID e SETGID

- Nota: è necessario che un utente con diritti di amministratore abbia, in precedenza, reso SETUID `root` una BASH
  - Ciò è possibile solo se si è in possesso delle credenziali di `root`

- Solo se questa condizione è verificata è possibile l'elevazione dei privilegi tramite

```
bash –p
```

Inibire SETUID e SETGID

- La copia di un file SETUID/SETGID perde il bit SETUID/SETGID
- Se così non fosse si potrebbe copiare un binario SETUID in una cartella locale e modificarlo a piacimento
  - Magari permettendo l'esecuzione di una shell

Inibire SETUID e SETGID

- Ad esempio, si copi il file `/usr/bin/passwd` in una directory locale

  ```
  cp /usr/bin/passwd .
  ```

- Se ne esaminino i permessi

  ```
  ls –l passwd
  ```

- Si noti che, in corrispondenza dei bit SETUID e SETGID, le `s` sono sostituite dalle `x`

```
-rwxr-xr-x 1 masucci masucci 54256 mar 10 18:29 passwd
```

Gestione dei privilegi negli altri linguaggi

- Tutti i meccanismi visti finora sono stati concepiti per l'uso in programmi scritti in linguaggio C
- E' possibile abbassare e ripristinare i privilegi in altri ambienti di programmazione?

Gestione dei privilegi negli altri linguaggi

- Nei linguaggi di scripting dinamici, tipo Python, i bit SETUID/SETGID sono ignorati negli script di ogni genere
- Si consideri lo scritpt `drop_rest.py` illustrato nella slide successiva
- Lo script prova ad effettuare abbassamento e ripristino di privilegi mediante funzioni di libreria

Gestione dei privilegi negli altri linguaggi

drop_rest.py

```
#!/usr/bin/python3

import os

(uid, euid, suid) = os.getresuid()
print("Prima dell'abbassamento privilegi:
        (uid, euid, suid) = ", (uid, euid, suid))

os.setresuid(-1, uid, -1)
(uid, euid, suid) = os.getresuid()
print("Dopo l'abbassamento privilegi:
        (uid, euid, suid) = ", (uid, euid, suid))

os.setresuid(-1, suid, -1)
(uid, euid, suid) = os.getresuid()
print("Dopo il ripristino privilegi:
        (uid, euid, suid) = ", (uid, euid, suid))
```

Gestione dei privilegi negli altri linguaggi

- Si consideri lo scritpt `drop_rest.py` illustrato nella slide precedente

- Lo si renda SETUID `root`

```
sudo chown root drop_rest.py
sudo chmod u+s drop_rest.py
```

- Si esegua lo script `drop_rest.py` con l'interprete Python standard

```
python3 drop_rest.py
```

Gestione dei privilegi negli altri linguaggi

- L'esecuzione è identica a quella in assenza di SETUID `root`
  - Lo script `drop_rest.py` stampa RUID, EUID, SUID dell'utente (normale) che lo ha lanciato
  - Non vi è traccia di ID privilegiati
  - Quindi il bit SETUID è stato ignorato

Gestione dei privilegi negli altri linguaggi

- Nei linguaggi di scripting dinamici, tipo Python, i bit SETUID/SETGID sono ignorati negli script di ogni genere

- L'unico programma che può impostare SETUID `root` è proprio l'interprete
  - Bisogna essere `root` per farlo
  - Avendo a disposizione un interprete SETUID `root` è possibile far gestire il privilegio all'interno dello script, tramite le funzioni di libreria fornite con l'interprete

Gestione dei privilegi negli altri linguaggi

- Ad esempio, si copi l'interprete Python in una directory locale

```
cp /usr/bin/python3 .
```

- Si renda SETUID `root` l'interprete

```
sudo chown root python3
sudo chmod u+s python3
```

- Si esegua lo script `drop_rest.py`

```
./python3 drop_rest.py
```

Gestione dei privilegi negli altri linguaggi

- Adesso l'abbassamento dei privilegi con lo script `drop_rest.py` funziona
- Ciò è dovuto al fatto che l'interprete Python è SETUID `root`

Gestione dei privilegi negli altri linguaggi

- Infine, l'ambiente di esecuzione di Java non supporta la gestione dei privilegi via SETUID/SETGID