Usare gli Hard-Link in ambienti Unix

Vi sono molte tecniche che aiutano gli attaccanti a sfruttare le debolezze dei sistemi, molte di queste sono veramente semplici anche se alle volte poco intuitive. Uno dei metodi ben noti ma sempre pericolosi è l'uso degli *hard-link* in ambiente Unix/Linux. Un *hard-link* è fondamentalmente un altro nome per lo stesso file, in pratica vi sono due nomi che puntano direttamente alla stessa area su disco, in termini tecnici allo stesso *inode*. Questo è da contrapporre ad un *soft-link* che è un nome che punta ad un altro nome, e solo quest'ultimo punta all'*inode* del file sul disco.

Per capire meglio la differenza, facciamo un esempio esplicito:[1]

```
> touch provahl
> ln provahl provahl2
> ln -s provahl provasl
> ls -il prova*

 33373 -rw-rw-r--   2 user user   0 Nov 26 10:56 provahl
 33373 -rw-rw-r--   2 user user   0 Nov 26 10:56 provahl2
 33484 lrwxrwxrwx   1 user user   7 Nov 26 10:56 provasl -> provahl
```

Con il comando `touch` abbiamo creato un file vuoto, di 0 byte, di nome `provahl`, con il comando `ln` abbiamo creato un *hard-link* di nome `provahl2` e con il comando `ln -s` un *soft-link* di nome `provasl`. Nella prima colonna del comando ls vediamo il numero di *inode* del file, praticamente della sua posizione fisica sul disco, nella terza colonna il numero di *hard-link*, ovvero di nomi, verso l'*inode* indicato, e nella sesta colonna la dimensione del file. Come si vede, il *soft-link* è un file su disco di 7 byte con un proprio *inode* che rimanda al file originario, mentre l'*hard-link* è indistinguibile dal file originario.

In tutte le implementazioni esistenti[2] degli *hard-link*, non è possibile fare un *hard-link* ad una intera directory e gli *hard-link* non possono attraversare i *filesystem* (partizioni), cioè se ad esempio `/var` e `/home` sono su *filesystem* diversi non si può avere che `/var/provahl` e `/home/provahl2` siano lo stesso file. Un'altra proprietà molto importante degli *hard-link* è che cancellando il nome di un *hard-link* (`/bin/rm provahl`) non si cancella, o meglio *un-link*, il file dal disco ma solo l'*hard-link* indicato, in altre parole `provahl2` continua a puntare allo stesso *inode* ed ad essere utilizzabile. Solo quando tutti gli *hard-link* ad un file sono stati rimossi, il file su disco non è più indirizzabile e l'area di disco può essere riutilizzata per nuovi file. Si noti che nel caso di *soft-link*, la rimozione del vero file (`provahl`) rende il *soft-link* inutilizzabile, in quanto quest'ultimo punta al

1 In questo articolo le opzioni dei comandi si riferiscono alle *GNU file utilities*, potrebbero variare in altre versioni.
2 Anche se questo non è richiesto dalla standard POSIX.

nome del file e non direttamente al suo *inode*:

```
> ls -il prova*

 33373 -rw-rw-r--   1 user user   0 Nov 26 10:56 provahl2
 33484 lrwxrwxrwx   1 user user   7 Nov 26 10:56 provasl -> provahl

> cat provasl
cat: provasl: No such file or directory
```

In questo caso il messaggio `No such file or directory` si riferisce al fatto che il file `provahl` non esiste più.

Come può un attaccante sfruttare gli *hard-link*? Supponiamo che su di un server Unix/Linux vi sia un programma *suid-root*, ovvero che sia eseguito da qualunque utente con i privilegi di amministratore del sistema, e che questo programma abbia ad esempio un buffer-overflow che permette ad un attaccante di ottenere localmente un accesso shell come l'utente *root* (l'amministratore del sistema). L'attaccante può quindi accedere alla macchina come un utente autorizzato e poi diventare amministratore senza che il vero amministratore se ne accorga dai log di accesso.

Per garantirsi in futuro l'accesso come amministratore, l'attaccante può creare un *hard-link* all'eseguibile che ha il buffer-overflow, ad esempio dando i comandi

```
> ln /usr/sbin/prog .myprog
> ls -il /usr/sbin/myprog .myprog
376345 -rwsr-xr-x  2 root root  290311  Jul 16 2002 .myprog
376345 -rwsr-xr-x  2 root root  290311  Jul 16 2002 /usr/sbin/prog
```

Ovviamente questo è possibile solo se l'attaccante può scrivere nella stessa partizione in cui è il programma originario (cosa che è sempre possibile all'amministratore del sistema).

Supponiamo ora che il vero amministratore decida di fare un upgrade del programma o di rimuoverlo del tutto. Ovviamente l'amministratore rimuoverà il file `/usr/sbin/prog` o lo sostituirà con un altro senza il buffer-overflow. Se il programma viene sostituito, avrà lo stesso nome ma un *inode* diverso in quanto l'*inode* 376345 è ancora utilizzato dal file `myprog`. All'attaccante questo però non interessa molto, poiché può sempre utilizzare il file `.myprog` e sfruttarne il buffer-overflow. Ad esempio, dopo l'upgrade potremmo avere la seguente situazione

```
> ls -il /usr/sbin/myprog .myprog
376345 -rwsr-xr-x  1 root root  290311  Jul 16 2002 .myprog
392458 -rwsr-xr-x  1 root root  296456  Nov 26 2003 /usr/sbin/prog
```

In alcuni casi è noto che attaccanti hanno creato preventivamente *hard-link* a molti (se non tutti) i programmi *suid-root* installati sulla macchina nella speranza che fosse scoperto successivamente un buffer-overflow in uno di essi. Se l'attaccante ha accesso solo come un utente normale, questo è possibile solo all'interno delle partizioni nelle quali vi sono directory in cui l'utente può scrivere.

Come può difendersi l'amministratore? Ovviamente una volta che il vero nome del programma con il buffer-overflow è stato cancellato o sostituito è difficile per l'amministratore capire se vi è un *hard-link* illegale alla vecchia versione. L'unica speranza è di trovare file con nomi ed in posizioni sospette ed esaminarli ad esempio con comandi quali *strings*, *ldd* ecc. per cercare di capire cosa sono.

L'azione che l'amministratore deve prendere è preventiva. **Prima** di rimuovere un programma o di farne l'upgrade, l'amministratore dovrebbe dare un comando del tipo[3]

```
> find /mydir \( -not -links 1 -a -not -type d \) -ls | sort | less

33373  0 -rw-rw-r--    2 user user  0 Nov 26 10:56 /mydir/provahl
33373  0 -rw-rw-r--    2 user user  0 Nov 26 10:56 /mydir/provahl2
```

nella partizione in cui è installato il programma. In questo modo ci si rende subito conto se vi sono *hard-link* illegali installati. Se si fa l'upgrade di un solo programma, è più semplice controllare manualmente l'*hard-link count* (terza colonna nell'output del comando `ls`, quarta colonna nell'output del comando `find`) dei file e nel caso di dubbi cercare all'interno della stessa partizione gli altri hard-link allo stesso programma, ad esempio con un comando del tipo

```
> find /mydir \( -not -links 1 -a -not -type d \) -ls | grep 33373

33373  0 -rw-rw-r--    2 user user  0 Nov 26 10:56 /mydir/provahl
33373  0 -rw-rw-r--    2 user user  0 Nov 26 10:56 /mydir/provahl2
```

Meglio ancora, l'amministratore può eseguire quotidianamente i comandi

```
> find /mydir \( -not -links 1 -a -not -type d \) -ls \
    | sort > hard-links-`date +"%Y%m%d"`
> diff hard-links-`date +"%Y%m%d"` \
    hard-links-`date --date='1 days ago' +"%Y%m%d"`
```

In questo modo ogni giorno viene fatta una lista di tutti gli *hard-link* multipli e viene confrontata con la lista del giorno precedente, identificando immediatamente i nuovi *hard-link* creati nelle ultime 24 ore.

Infine, ma dovrebbe essere la prima cosa da fare, ci si può proteggere preventivamente progettando con cautela il partizionamento dei dischi. Poiché gli *hard-link* non possono attraversare le

3 Per una maggiore leggibilità dell'output, si può sostituire il `less` finale con `awk '{if ($1 != n) {print "\n"$0; n=$1} else {print $0}}' n="0" - | less`

partizioni, conviene dedicare ad `/home/`, `/tmp/` e `/var/` partizioni indipendenti,[4] essendo queste le *top-directory* alle quali hanno di solito accesso in scrittura gli utenti e che non contengono eseguibili di sistema. In generale bisognerebbe progettare il sistema in modo che nelle partizioni in cui vi sono gli eseguibili di sistema gli utenti non abbiano accesso in scrittura, e che nelle partizioni in cui gli utenti hanno accesso in scrittura non vi siano eseguibili di sistema. Come sempre, in caso di infrazione della macchina, la cosa più sicura dopo un'attenta analisi di quello che è successo in modo da prevenirlo nel futuro, è reinstallare la macchina riformattando tutti i dischi.

Andrea Pasquinucci
Libero Professionista in Sicurezza Informatica
pasquinucci@ucci.it

4 Per ridurre il numero di partizioni, `/tmp` è spesso un *soft-link* a `/var/tmp`.