Rootkit: Analysis, Detection and Protection

Igor Neri

1 Definizione

Un rootkit è una raccolta di tool che un intruso installa sulla macchina vittima dopo essersi guadagnato un accesso non autorizzato alla stessa. Lo scopo principale di un rootkit è identico a quello di un trojan, cioè quello di permettere all'intruso (o agli intrusi) di poter ritornare al sistema compromesso senza essere scoperti.
In particolare un rootkit non garantisce l'accesso al sistema, che deve essere guadagnato in altro modo, ma permette principalmente di:

- Nascondere l'attività dell'intruso
- Garantire un accesso non autorizzato (facoltativo)
- Rimuovere le tracce dai log di sistema

2 Classificazione

I rootkit si classificano principalmente in base al livello al quale operano, i più diffusi S.O. per le architetture x86 utilizzano solo due dei quattro livelli di privilegi disponibili (fig.1):

- il ring 0, corrispondente al livello kernel, dove c'è pieno accesso alla memoria e all'intero set di istruzioni.
- il ring 3, corrispondente al livello utente, che prevede accesso limitato alla memoria ed ad un set di istruzioni ridotto.

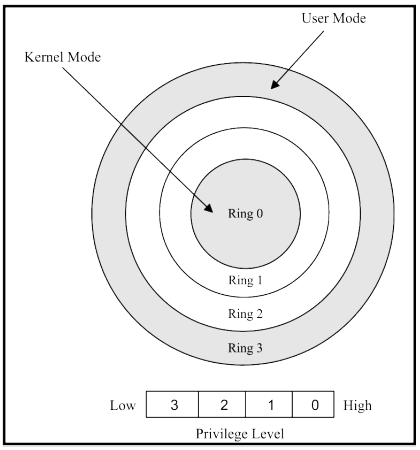


Figura 1: x86 Rings

Considereremo quindi due principali categorie di rootkit, quelli a livello utente (userspace) e quelli a livello kernel (kernelspace), tuttavia con l'emergente uso delle macchive virtuali c'è stato un crescente interesse nello svoluppo di rootkit a livello hypervisor, che non tratteremo in questo seminario.

2.1 Userspace

I rootkit a livello utente prevedono la modifica di binari di sistema e/o di librerie per modificarne il comportamento permettendo di nascondere le attività dell'attaccante. Come abbiamo visto uno degli obiettivi principali dei rootkit è *nascondere*, quindi il rootkit dovrà rimpiazzare i binari di sistema per nascondere:

- File: du, find, sync, ls, df, lsof, netstat
- Processi: killall, pidof, ps, top, lsof
- Connessioni: netstat, tcpd, lsof, route, arp
- Log entry: syslogd, tcpd
- Login: w, who, last

Una volta che l'attaccante si è nascosto nel sistema vuole comunque garantirsi la possibilità di riaccedere al sistema in modo diretto, quindi compito (opzionale) del rootkit è garantire l'accesso al sistema inserendo ad esempio una backdoor nei processi che vengono eseguiti in fase di boot del sistema o comunque in applicazioni normalmente destiante a garantire accesso autorizzato al sistema modificandole. Ad esempio modificando i programmi: inetd, login, rlogin, rshd, telnetd, sshd, su, chfn, passwd, chsh, sudo.
Fatto ciò deve nascondere le tracce della violazione e quindi usare appositi strumenti che gli permettano di eliminare le entry relative all'accesso non autorizzato dai log di sistema.
Come è facile intuire l'installazione di questo tipo di rootkit risulta abbastanza laboriosa, principalmente perchè è necessario sostituire gran parte degli eseguibili del sisitema manatendone comunque la compatibilità con il sistema stesso.

2.2 Kernelspace

I rootkit che operano a livello kernel cercano di ovviare al poblema dei rootkit a livello utente andando a modificare il comportamente delle chiamate di sistema (syscall). L'obiettivo è quindi quello di piazzare il codice maligno al livello kernel andandone a modificare la struttura.
Prima di spiegare come funzionano dobbiamo vedere cosa succede quando viene eseguita un'applicazione che fa uso di chiamate di sistema.

2.2.1 Flusso di esecuzione di chiamata ad una syscall

Nell'esempio in fig.2 vediamo come opera il comando *ls*, durante l'esecuzione c'è una chiamata alla funzione *getdents()* che legge le informazioni relative alla directory passata come argomento, la funzione *getdents*, attraverso un interfaccia utente-kernel fa riferimento alla syscall *sys_getdents()*. L'obiettivo dei rootkit a livello kernel è di intercettare il flusso di esecuzione per modificarlo, questo può essere fatto a vari livelli.

2.2.2 Il normale Workflow

Vediamo più in dettaglio cosa succede quando viene chiamata una syscall (fig.3):

- Il processo chiamante piazza un interrupt della IDT (interrupt desctiption table)

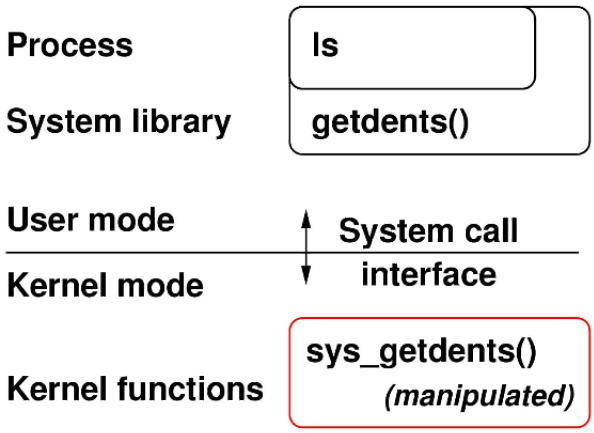


Figura 2: Workflow del comando *ls*

- L'interrupt handler consulta la IDT e richiama il system call handler
- Il system call handler, in base alla funzione richiesta preleva il puntatore alla syscall specifica dalla system call table.
- La system call viene eseguita, facendo anche riferimento ad altre funzioni del kernel

Il rootkit può essere inserito in vari livelli dirottando il flusso di esecuzione.

2.2.3 Manipolare la Syscall Table

Uno dei primi metodi è quello di modificare la syscall table (fig.4(a)) andando a modificare il puntatore alla syscall da eseguire sostituendolo con il puntatore ad una modificata. La nuova funzione viene quindi chiamata al posto di quella originale, spesso la nuova funzione fa soltanto da wrapper alla vecchia aggiunendo dei controlli.

2.2.4 Copia della syscall table/handler

Uno dei modi più semplici per rilevare un rootkit del tipo precedente è fare una copia della systemcall table originale e confrontarla periodicamente con quella attuale. Per ovviare a questo problema possiamo farci una copia della syscall table e modificare il syscall handler per fargli fare riferimento alla syscall table modificata (fig.4(b)).

2.2.5 Manipolare la IDT

Un'altro punto per intercettare il flusso di esecuzione è modificare la IDT (fig.4(c)) modificando il puntatore alla syscall handler facendogli chiamare uno modificato che si occuperò di eseguire il rootkit.

2.2.6 Modificare la syscall all'interno del kernel

Per evitare di modificare la struttura precedente possiamo modificare direttamente la funzione chiamata dalla syscall per alterarne il comportamento (fig.4(d)), questo metodo risulta tuttavia difficile da applicare perchè richiede spesso una ricompilazione del kernel e un riavvio della macchina.

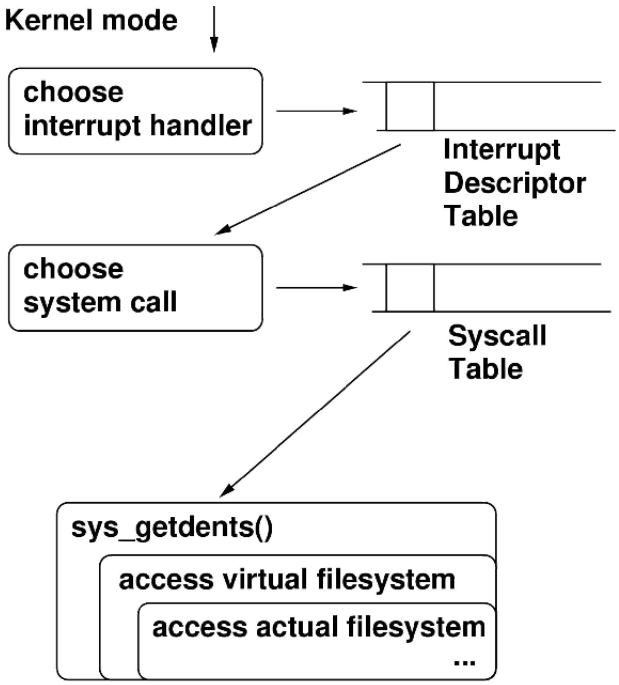


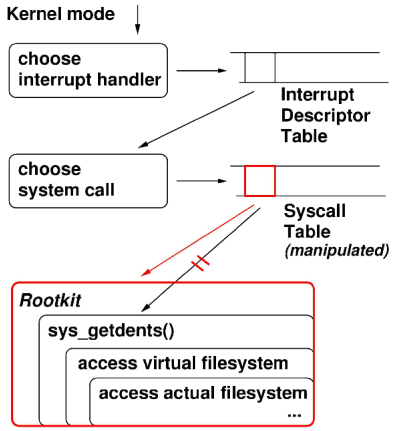
Figura 3: Flusso di esecuzione

3 Rilevare la presenza di rootkit

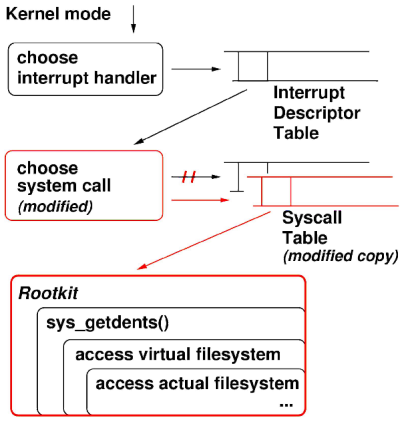
I metodi di rilevamento cambiano molto in base al tipo di rootkit utilizzati. Per il primo tipo di rootkit, quelli a livello utente, i metodo più efficaci sono un'analisi dei binari di sistema cercando di identificare modifiche non autorizzata (checksum) o cercando di rilevare le *signature* dei rootkit comosciuti all'interno dei binari, ovviamente nuovi rootkit sconosciuti non saranno identificabili con quest'ultimo metodo.
Per identificare rootkit a livello kernel invece possiamo affidarci a vari strumenti che vanno dall'identificare modifiche nella memoria destinata al kernel (kstat), all'uso di specifici moduli per il kernel che hanno come obiettivo quello di restringere e monitorare le azioni possibili a livello kernel.
Metodi che funzionano in entrambi i casi si basano invece in un analisi offline del sistema che si presume sia compromesso o da un terzo nodo della rete per identificare comportamente sospetti.

4 Protezione del sistema

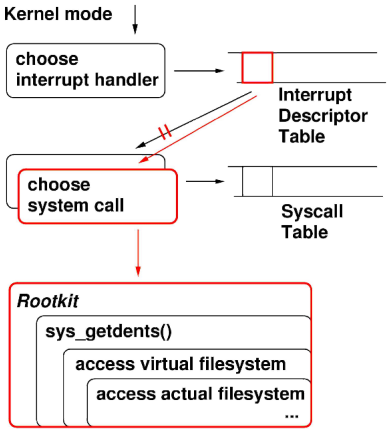
La prevenzione è un aspetto importante, è necessario assicurarsi che il prorpio sistema sia al sicuro alemno dalle minacce conosciute, e comunque che i comportamenti sospetti o non autorizzati siano impediti. Applicando delle semplici linee guida di sicurezza, che vanno dall'applicare periodicamente le patch di sicurezza al restingere le operazioni possibili, possiamo considerare il nostro sistema sufficientemente sicuro.



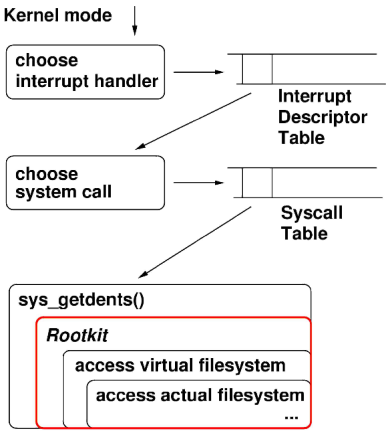
(a) Manipolare la Syscall Table



(b) Copia della syscall table/handler



(c) Manipolare la IDT



(d) Modificare la syscall all'interno del kernel

Figura 4: N-1 Performance