Free & Secure

Alcune feature di sicurezza nei sistemi operativi free

Guido Bolognesi
guido@kill-9.it

Sommario

- Il controllo degli accessi (MAC e DAC, RSBAC)
- La gestione della memoria,
- disco,
- rete
- Alcune implementazioni (linux, *BSD)

Una Domanda

Qual è il più grande
problema di sicurezza dei sistemi
operativi UNIX-like?

Risposta

Qual è il più grande
problema di sicurezza dei sistemi
operativi UNIX-like?

root.

Il modello classico di sicurezza

POSIX.1e (IEEE 1003.1e)

Definisce il modo di suddividere i privilegi di root.

- 1003.1e *Security APIs*
- 1003.2c *Security Commands and Utilities*

Purtroppo il 15 Gennaio 1998,
dopo 13 anni di sviluppo,
IEEE ha ritirato il suo supporto.

POSIX.1e (IEEE 1003.1e)

Nell'implementazione di Linux ci sono 28 *capabilities*
Sono previste sia per i permessi "classici":

- `KILL` Per mandare un segnale ad un processo
- `SETSID` Permette la chiamata delle `*suid()`

che per una divisione delle funzionalità più granulare

POSIX.1e (IEEE 1003.1e)

- `NET_ADMIN` modificare la configurazione di rete
- `NET_RAW` per accesso alle raw socket
- `SYS_PTRACE` utilizzo della `ptrace()`
- `SYS_PACCT` modificare l'accounting di un processo
- `SYS_NICE` modificare la priorità di processi anche non dell'utente
- `LINUX_INSMOD` `LINUX_RMMOD` per la gestione dei moduli

MAC vs. DAC

MAC definisce formalmente un modello per

“ridurre l’accesso a degli oggetti basandosi sulla sensibilità dell’informazione contenuta, e l’autorizzazione formale a dei soggetti ad accedere ad informazioni di quella sensibilità”

(Dipartimento della Difesa, Trusted Computer System Evaluation Criteria)

MAC vs. DAC

L'opposto é il DAC (Discretionary access control)

"un modo per ridurre l'accesso agli oggetti in base all'identità del soggetto o al gruppo a cui appartiene"

Differenza: un soggetto del DAC puó direttamente o indirettamente passare i propri privilegi ad un altro soggetto.

RSBAC

Rule Set Based Access Control

É un framework per Linux esteso e completo che viene utilizzato per implementare diversi modelli di sicurezza all'interno di un sistema.

Basato su GFAC (Generalized Framework for Access Control) di Abrams e LaPadula.

`system call` $\rightarrow$ componente di decisione centrale $\rightarrow$ {moduli decisionali attivi} $\Rightarrow$ decisione composta.

MAC (B-LP)

Mandatory Access Control
(modello Bell-LaPadula)

- Subjects S, Objects O
- A = [*execute, read, write, read-write*]
- x é un modo di A
- ogni (S_i, O_j, x) definisce l'insieme l'accesso corrente
- viene definita una gerarchia H (padre-figlio, uno o piú alberi indipendenti)

MAC (B-LP)

- ogni $M_{i,j}$ definisce gli accessi autorizzati di S_i a O_j

Il Security level é definito come

SL = (sec. classification S_c *, category* C*)*

con, ad esempio

S_c = *public, confidential, secret, top-secret*

e la category come assegnazione formale ad un'area.

MAC (B-LP)

Si dice che una entitá con security level (S_{c1}, Cat_1) domina (S_{c2}, Cat_2) quando

$$S_{c1} >= S_{c2} \text{ e } Cat_1 \text{ contiene } Cat_2.$$

Infine, la funzione di classificazione F é definita come $F(f_s, f_o, f_c)$

- $f_s(S_i)$ massimo livello di sicurezza di S_i
- $f_o(O_j)$ livello di sicurezza di O_j
- $f_c(S_i)$ livello di sicurezza attuale di S_i

Uno stato del sistema é $z_k(b, M, F, H)$

Stack non eseguibile

Marcare lo stack come non eseguibile elimina i buffer overflow che iniettano codice nello stack e poi lo eseguono, peró:

- Linux usa parte dello user stack per la gestione dei segnali
- gcc lo utilizza per le funzioni innestate (trampoline, OBSD-propolice)
- alcuni programmi funzionali lo utilizzano per la generazione di codice a runtime
- ... non tutti gli exploit fanno injection sullo stack!

L’uso di $RANDOM

La chiave di molti exploit é l’utilizzo del bruteforce, quindi si tenta di impedire o rendere molto piú difficoltosa la tecnica.

Alcuni metodi di prevenzione sono assolutamente compatibili con gli standard.

Un moderno S.O. sicuro dovrebbe randomizzare almeno:

- alcuni parametri dello stack TCP/IP
- l’allocazione della memoria

L’accesso alla memoria

É utile randomizzare:

- il *base pointer* per lo stack
- *`mmap()` base*
- *executable base*

In piú, PID random non fanno male.

L’accesso al disco

Alcuni problemi sono giá noti, altri lo diventeranno:

- `mktemp(3)`, `tmpnam(3)` potrebbero dare problemi (ispell, cron, popd)
- usare `mkstemp(3)`, `tempnam(3)`, `tmpfile(3)`
- OpenBSD randomizza gli inode
- stare attenti a [sym|hard]link (/tmp races, anyone?)

L’accesso alla rete

Randomizzare é bello:

- source port delle connessioni uscenti
- IP ID (a volte offset costanti... e se offset=0?)
- (Time To Live, entro certi limiti)

Sono tutti provvedimenti utili per il fingerprinting (e a volte altro) remoto.

É ovvio che se li implementano un numero piccolissimo di S.O. ...

Implementazioni

Diamo uno sguardo ad alcune implementazioni di quello che abbiamo visto:

- grsecurity
- SElinux
- TrustedBSD
- OpenBSD
- (systrace)

Grsecurity

Nasce come port della patch di OpenWall alla serie 2.4 dei kernel linux

- non executable stack
- pagine di memoria non eseguibili (PaX)
- gcc trampoline emulation
- implementa RBAC (role based access control)
- randomize PID, source port, TTL, IP ID

Grsecurity (cont.)

- randomize stack, mmap base, executable base
- /proc non leggibile
- protezione dalle fork-bomb
- fork logging, set[ui|gi]ds
- chroot evoluta (no signaling, no [u]mount, no mknod, no sysctl, exec logging)
- restrizioni su symlink/hardlink (/tmp races)

SElinux

Sviluppato dalla NSA (National Security Agency) americana, in collaborazione con i NAI labs, la SCC (Security Computing Corp.) e MITRE Corp.

- kernel di linux la cui funzione primaria e' implementare Mandatory Access Control
- supporta le feature standard (file, socket, messages...)
- rilasciato in GPL senza export limitations

Previste: integrazione IPSec, MAC su NFS.
Al momento non accettano contributi...

TrustedBSD

Sponsorizzato da DARPA (Defence Advanced Research Projects Agency), NSA e Network Associates Laboratories.
Propone:

- ACL (come attributo su FFS, nativo su UFS2 – già in FreeBSD 5.x-CURRENT)
- Event Auditing (in sviluppo, integrazione prevista 2003Q3)
- Extended Attributes (integrazione dei dati aggiuntivi per ACL, MAC e capabilities sul filesystem, già in 5.0-CURRENT)

TrustedBSD (cont.)

- Fine-grained capabilities, per consentire a processi non di root di accedere a risorse sensibili (implementazione funzionante ma non ancora completa, in integrazione)
- MAC implementato di base (per problemi tipici, visibilità reciproca degli utenti), NA (CBOSS Project) sta portando il framework di SELinux come un'estensione, SEBSD buona parte in backport nel tree di FreeBSD 5.0-RELEASE

OpenBSD

Tradizionalmente orientato alla security.
“Secure by default”, total code audit, crittografia integrata (kernel + userland)

Utilizza PRNG (Pseudo random number generator), “nutrito” da:

- interrupt timing
- network data interrupt latency
- inter-keypress timing
- informazioni sull’I/O dischi

OpenBSD (cont.)

Con PRNG randomizza:

- `bind()` delle porte
- PID dei processi
- IP ID
- DNS query ID
- numero della generazione degli inode
- nomi file temporanei per `mkstemp()`
- NFS RPC transaction ID

Offre implementazione nativa di `systrace`

systrace

Niels Provos (OpenSSH, honeyd)

- permette di definire quali *system call* sono permesse alle applicazioni e privilege elevation (meno suid)
- l'utente puó definire una policy
- generazione automatica, poi tuning
- riscrittura degli argomenti di una syscall (chroot, format overflow)
- genera warning via syslog, quindi anche monitor di macchine remote.

`systrace` (cont.)

Un esempio:

```
Policy:  /bin/ls, Emulation:  native
[...]
native-fsread:  filename eq /tmp then permit
native-fsread:  filename eq /etc then
deny[enotdir]
[...]
```

Ricordarsi che quello che non é esplicitamente permesso viene negato!

debian package, nativo Net/OpenBSD, port vecchio per Free, MacOSX

Angel

Progetto italiano (Paolo “sponge” Perego, Aldo Scaccabarozzi)
Un kernel module che

- controlla il traffico uscente da un host (syn flood, spoofing, jolt, Outlook BOF)
- alcuni attacchi locali (format string, fork/malloc bombing, BOF contro suid)

Ha persino una applet per WindowMaker...
http://www.sikurezza.org/angel

Q&A, Ringraziamenti

- vi, LaTeX, dia e la comunità OpenSource
- Koba, Fusys
- Laura