

Introduzione al TE/DTE

a cura di Giorgio Zanin

Sistemi di Elaborazione: sicurezza

a.a. 2002-2003

Introduzione

- Attualmente la sicurezza di Unix risiede in:
 - Bit di protezione
 - Utente root
 - Meccanismi setuid/setgid
- Grande responsabilita' riposta negli amministratori di sistema e nelle applicazioni
- E' necessario assumere che le applicazioni privilegiate possano essere vulnerabili ad attacchi
- La compromissione di un sottosistema privilegiato (lpd, fingerd, rdist,...) puo' rendere vulnerabile tutto il sistema
- L'analisi delle applicazioni e' improponibile e non da' garanzie
- Unix ed altri sistemi operativi possono, in teoria, essere resi maggiormente sicuri con un altro layer per il CONTROLLO DEGLI ACCESSI

- Diversi modelli di controllo degli accessi sono stati proposti:
 - Bell-LaPadula
 - Biba
 - MLS
 - TE
 - RBAC
 - ...
- Molto successo teorico, poco successo pratico
- I modelli implicano:
 - Costi significativi
 - Amministrazione piu' complessa dei sistemi
 - Incompatibilita' delle applicazioni
 - Addestramento degli utenti

Type Enforcement

- Formalizzato da Boebert e Kain (1985)
- ACM mandatorio orientato alle tabelle
- Come molti ACMs:
 - Entita' attive (Soggetti)
 - Entita' passive (Oggetti)
- Dominio: associato ai soggetti (processi)
- Tipo: associato agli oggetti (file, messaggi, segmenti di memoria condivisa)
- Domain Definition Table (DDT): definisce le modalita' di accesso consentite tra domini e tipi
- Domain Interaction Table (DIT): definisce le modalita' di accesso consentite tra domini
- All'esecuzione del sistema si effettuano lookup nelle tabelle per verificare l'esistenza dei permessi necessari
- Viene usato per confinare le applicazioni e restringere i flussi di informazioni

DDT

- Interazioni Domini-Tipi = Soggetti-Oggetti
- Righe: Domini – Colonne: Tipi
- Quando un soggetto S cerca di accedere ad un oggetto O si selezionano la riga del dominio di S e la colonna del tipo di O

DIT

- Interazioni Domini-Domini
- Regole analoghe a DDT

Domain and Type Enforcement

- Evoluzione del TE
- La configurazione tabellare del TE e' troppo complessa
- Le tabelle non mappano naturalmente le strutture standard del sistema (gerarchie per file e processi)
- Non vengono utilizzate esplicitamente le tabelle (troppo difficili da gestire)
- Le policy vengono specificate in un linguaggio di alto livello (DTEL)
- Si utilizza il typing implicito per assegnare i tipi ai file
- Tutti i processi, anche quelli root, sono soggetti ai controlli DTE
- E' possibile soddisfare diversi requisiti di sicurezza e diverse policy

DTEL

- DTEL fornisce 4 costrutti principali per esprimere una policy DTE:
 - **type**: dichiara uno o più tipi di oggetti da usare in altre parti della specifica; definisce delle classi di equivalenza di dati
 - **domain**: definisce l'ambiente di esecuzione ristretto per i processi
 - **initial_domain**: definisce il dominio iniziale (quello per il primo processo, `init` per Unix)
 - **assign**: associa i tipi ai file

domain

- E' espresso come una lista di tuple ed e' composto di tre parti:
 - entrypoint: programmi che devono essere eseguiti da un processo per entrare nel dominio; sono identificati da un nome completo di percorso
 - diritti di accesso a tipi di oggetti (rwx->foo_t)
 - diritti di accesso a soggetti in altri domini (sigkill->user_d)
- Diritti specificabili sugli oggetti: r, w, x, c, d
- Diritti specificabili sui soggetti: segnali, auto, exec
- Nel DTE, DDT e DIT collassano in un unico costrutto
- E' possibile specificare delle transizioni di dominio:
 - Automatiche: tramite la modalita' di accesso *auto*
 - Esplicite: tramite la modalita' di accesso *exec*

Transizioni di dominio

■ **auto**

se un dominio A ha modalita' di accesso *auto* su B, quando esegue (con una chiamata alla syscall `exec()`) un entrypoint di B, genera automaticamente un soggetto in B

Serve a mantenere compatibilita' con le applicazioni estranee al DTE: viene forzata una transizione automatica di tipo, senza modificare gli eseguibili delle applicazioni

■ **exec**

se un dominio A ha modalita' di accesso *exec* su B, un soggetto in A puo' creare un soggetto in B, eseguendo un entrypoint di B e richiedendo che il programma giri in B

assign

- Può essere ricorsivo: si applica alla directory e a tutto ciò che c'è sotto
- Può essere sovrascritto
- Realizza il *typing implicito*
- I tipi degli oggetti vengono assegnati in base alla gerarchia delle directory stabilendo regole generali ed elencando eccezioni
- In ogni sottodirectory i file hanno il tipo della sottodirectory
- Pochi tipi per molti file
- Il tipo dell'oggetto non cambia durante la sua vita
- Opzioni disponibili:
 - **r**(ecursive): si applica a tutti i percorsi con il dato prefisso
 - **s**(tatic): non permette di creare oggetti di tipi differenti a quello dato

initial_domain

- Una policy DTEL rispecchia la gerarchia dei processi
- Domini = attributi di processi ereditati per default da processo a processo
- `/etc/init` primo processo da cui discendono tutti gli altri
- Specificazione del dominio del primo processo
- I processi figli ereditano i domini dei genitori ed eventualmente migrano secondo le modalita' `exec` e `auto` all'invocazione della syscall `exec()`

Confinamento di programmi root con DTE

- L'uso pervasivo dei privilegi di root e' una delle maggiori sorgenti di problemi di sicurezza in Unix
- Un solo programma root sovvertito puo' far ottenere il controllo dell'intera macchina
- "Tutte le uova in un cesto"
- I programmi root sono estremamente attraenti per gli attaccanti
- Trovare e fissare le vulnerabilita' dei programmi root e' importante ma non basta
- Configurando una politica DTE e' possibile eseguire i programmi root in domini ristretti che permettano solamente accessi appropriati alle responsabilita' di ciascun programma: "le uova di Unix in cesti separati"
- *Principio dei privilegi minimi*
- Vengono tagliate via le possibilita' di accesso non necessarie alle mansioni dei programmi in questione
- DTE restringe i processi root e quelli non root ed opera in aggiunta ai bit di protezione di Unix

Strategia

- Speciale dominio amministrativo che ha accesso in scrittura ai binari di sistema e alle directory che li contengono
- La transizione in questo dominio avviene solo dopo l'autenticazione dell'utente e i controlli di autorizzazione
- Tutti gli altri processi, demoni root inclusi, vengono eseguiti in domini meno potenti (senza quell'accesso in scrittura)
- Se un attaccante sovrverte un demone root non può fare più di quello concesso dal dominio del demone
- Nei domini dei demoni non c'è il diritto di scrittura dei binari di sistema
- Attaccante non capace di sostituire i programmi che costituiscono le backdoor nascoste di Rootkit

Una semplice policy DTE

- Tutti i processi in 4 domini:
 - **daemon_d** per i demoni di sistema compreso `init`
 - **login_d** per il programma di login modificato
 - **user_d** per le sessioni utente ordinarie
 - **admin_t** per le sessioni dell'amministratore di sistema
- *daemon_d* transisce automaticamente in *login_d* all'invocazione di login da parte dell'apposito demone (`getty`)
- *login_d* puo' transire in *user_d* o *admin_d* a seconda del tipo di sessione che l'utente vuole iniziare

daemon_d

- Dominio nel quale gira il primo processo:
 - Costrutto `initial_domain`
 - Entrypoint del dominio: `/sbin/init`
- Tutti i discendenti di `init` girano in `daemon_d` fino a che `/usr/bin/login` viene invocato
- Unico modo per transire da questo ad un altro dominio: invocazione di `login`
- La transizione e' automatica

```
domain daemon_d = (/sbin/init),  
    (crwd->writable_t),  
    (rxd->binaries_t),  
    (rd->generic_t, readable_t),  
    (auto->login_d);
```

login_d

- E' l'unico dominio che puo' transire in user_d e in admin_d
- Non bypassabile
- E' possibile eseguire un solo binario: il suo entrypoint
- Manca il diritto di esecuzione di un qualsiasi tipo: negato ogni tentativo di eseguire un altro binario senza transire in un altro dominio (cambiando i privilegi)

```
domain login_d = (/usr/bin/login),  
    (crwd->writable_t),  
    (rd->generic_t, readable_t, dte_t),  
    (exec->user_d, admin_d);
```

user_d

- Per default ogni oggetto creato in questo dominio ha tipo `generic_t`
- Gli entrypoint di questo dominio sono le shell (`/usr/bin/{sh, csh, tcsh}`)
- Agli utenti non amministratori il sistema sembra comportarsi come un normale Unix
- Agli amministratori non vengono permesse le funzioni amministrative (neanche diventando superuser)

```
domain user_d = (/usr/bin/{sh, csh, tcsh}),  
    (crwxd->generic_t),  
    (rwd->writable_t),  
    (rxd->binaries_t),  
    (rd->readable_t, dte_t);
```

admin_d

- E' l'unico dominio in cui gli amministratori possono espletare le proprie funzioni con pieni poteri
- Si possono creare e modificare file binari e di configurazione di Unix
- Queste potenzialita' non sono presenti in nessun altro dominio e sono comunque possibili solo dopo l'autenticazione di login

```
domain admin_d = (/usr/bin/{sh, csh, tcsh}),  
    (crwxd->generic_t),  
    (rwx-d->writable_t, binaries_t, readable_t, dte_t);
```

- La precedente semplice policy e' sufficiente a non permettere allo script di installazione di Rootkit, invocato in una shell root in daemon_d e user_d, di sostituire i programmi di sistema
- Un maggior livello di sicurezza si puo' ottenere definendo opportuni domini per:
 - file delle password
 - memory device file (/dev/{mem,kmem})
 - disk device file (/dev/{sd0a,sd0b,...})
 - log di sistema
 - Web browser

```
type generic_t, binaries_t, dte_t, readable_t, writable_t;
```

```
domain daemon_d = (/sbin/init),  
    (crwd->writable_t),  
    (rxd->binaries_t),  
    (rd->generic_t, readable_t),  
    (auto->login_d);
```

```
domain login_d = (/usr/bin/login),  
    (crwd->writable_t),  
    (rd->generic_t, readable_t, dte_t),  
    (exec->user_d, admin_d);
```

```
domain user_d = (/usr/bin/{sh, csh, tcsh}),  
    (crwd->generic_t),  
    (rwd->writable_t),  
    (rxd->binaries_t),  
    (rd->readable_t, dte_t);
```

```
domain admin_d = (/usr/bin/{sh, csh, tcsh}),  
    (crwd->generic_t),  
    (rwd->writable_t, binaries_t, readable_t, dte_t);
```

```
initial_domain = daemon_d;
```

```
assign -r generic_t      /;  
assign -r writeble_t     /usr/var, /dev, /tmp;  
assign -r readable_t     /etc;  
assign -r -s dte_t       /dte;  
assign -r -s binaries_t  /sbin, /bin, /usr/{sbin, bin}, /usr/local/bin;
```

Bibliografia

- Boebert, Kain, “A practical Alternative to Hierarchical Integrity Policies”
- Badger, Sterne, Sherman, Walker, Haghghat, “A domain and type enforcement Unix Prototype”
- Badger, Sterne, Sherman, Walker, Haghghat, “Practical Domain and Type Enforcement for Unix”
- Walker, Sterne, Badger, Petkac, Shermann, Oostendorp, “Confining root programs with domain and type enforcement (DTE)”