

## 4 Domain Transition

G.5 Beispiel SELinux

- Standard-UNIX-Situation:
  - Programm benötigt zum Ablauf besondere Privilegien
  - s-Bit bewirkt Umschalten der `uid` im Rahmen des `execve`-Systemaufrufs
  - eigentlich sehr grob-granulare Lösung
    - alle oder viele (Gruppe) dürfen das Programm ausführen
    - das Programm hat - egal wer es ausführt - alle Rechte des Dateibesitzers
- SELinux erlaubt Verfeinerung der Fragestellung
  - wie bekommt der Prozess, der `/usr/bin/passwd` ausführt den Domain-Typ `passwd_t` ?
  - mehrere Teilprobleme:
    1. welche Domains dürfen `/usr/bin/passwd` ausführen?
    2. welche Prozesse (d. h. Instanzen welcher Programme) dürfen den Domain-Typ `passwd_t` bekommen?
    3. welche Domains dürfen bei Ausführung eines solchen Programms in den Domain `passwd_t` überführt werden?

SYSSEC

Systemsicherheit  
© • Universität Erlangen-Nürnberg • Informatik 4, 2008

G-Modelle\_fm 2008-01-31 09\_49

G.181

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

## 4 Domain Transition (3)

G.5 Beispiel SELinux

- explizit
  - vor einem `execve`-Aufruf wird der Security Context dafür gesetzt (`setexeccon(3)`)
  - Problem: aufrufendes Programm hat meist kein explizites Wissen über den besonderen Rechtebedarf des ausgeführten Programms: s-Bit-Semantik setzt `uid` automatisch um - aufrufendes Programm
- implizit : `process passwd_t`
  - spezielle Regel in der SELinux Policy
  - `type_transition user_t passwd_exec_t : process passwd_t;`  
wenn ein Prozess der Domain `user_t` ein Objekt mit Typ `passwd_exec_t` ausführt, wird die Domain des Prozesses automatisch nach `passwd_t` überführt. Die Regel bezieht sich auf die Objektklasse `process`
    - ein Prozess (normalerweise ein Subjekt) ist hier Objekt im Sinn der Regel - d. h. die Regel bezieht sich auf Prozesse

SYSSEC

Systemsicherheit  
© • Universität Erlangen-Nürnberg • Informatik 4, 2008

G-Modelle\_fm 2008-01-31 09\_49

G.183

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

## 4 Domain Transition (2)

G.5 Beispiel SELinux

- SELinux-Lösung:
  1. Ausführungsrecht für `/usr/bin/passwd`
    - "normale" Benutzerprozesse laufen in Domain `user_t`
    - `/usr/bin/passwd` hat Typ `passwd_exec_t`
    - `allow user_t passwd_exec_t : file (getattr execute);`  
"normale" Benutzerprozesse (Prozesse in Domain `user_t`) dürfen `stat(2)` und `execve(2)` auf `/usr/bin/passwd` ausführen
  2. "Potentielles s-Bit" für das Programm `/usr/bin/passwd`
    - `allow passwd_t passwd_exec_t : file entrypoint;`  
Prozesse, die das Programm `/usr/bin/passwd` ausführen, dürfen in Domain `passwd_t` überführt werden
    - strikte Kontrolle, welche Programme eine Domain betreten dürfen!
  3. s-Bit-Nutzungsrecht für Domain `user_t`
    - `allow user_t passwd_t : process transition;`  
Prozesse in Domain `user_t` dürfen in Domain `passwd_t` überführt werden

SYSSEC

Systemsicherheit  
© • Universität Erlangen-Nürnberg • Informatik 4, 2008

G-Modelle\_fm 2008-01-31 09\_49

G.182

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

## 5 Rollen

G.5 Beispiel SELinux

- Mechanismus zur Einschränkung von Domain-Zuordnungen oder Domain-Transitionen
  - Gegensatz zu RBAC-Modellen, die Rollen mit Rechten verknüpfen!
  - Rechte in SELinux ergeben sich nur aus *Type-Enforcement-Mechanismus*
  - Rollen schränken Rechte ein
- Zuordnung von Linux-Benutzern (`uid`) zu Rollen
  - `user joe roles { user_r, system_r };`  
Prozesse des Benutzers `joe` dürfen die Rollen `user_r` und `system_r` einnehmen
- Kompatibilität von Domain-Typen und Rollen
  - `role user_r types user_t;`  
`role user_r types passwd_t;`  
die Domains `user_t` und `passwd_t` sind mit der Rolle `user_r` kompatibel  
=> Prozesse in der Rolle `user_r` dürfen eine Domain-Transition zwischen `user_t` und `passwd_t` durchführen

SYSSEC

Systemsicherheit  
© • Universität Erlangen-Nürnberg • Informatik 4, 2008

G-Modelle\_fm 2008-01-31 09\_49

G.184

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

## 5 Rollen (2)

G.5 Beispiel SELinux

### ■ Ziele

1. Entkopplung von Linux-Benutzer-Ids und SELinux-Domains
  - ▶ große Benutzerzahl, viele Domains
  - ▶ wenige "typische" Rollen fassen die Rechte auf die Typen zusammen ("normaler Benutzer", "Administrator", ...)
  - ▶ Zuordnung der Rollen zu den Benutzern überschaubar
2. Rechte eines Benutzers werden auf die gerade "aktive" Rolle beschränkt
  - ▶ jeder Prozess hat jeweils eine "aktive" Rolle
  - ▶ Domaintransitionen sind auf dazu kompatible Domains beschränkt
  - ▶ Wechsel der "aktiven" Rolle erlaubt temporären Rechtewechsel (ähnlich wie su/sudo-Kommando, aber feiner abstimmbare)

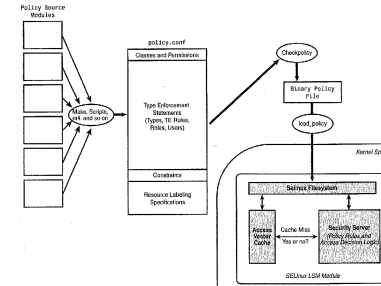
## 7 SELinux Kern-Architektur

G.5 Beispiel SELinux

### ■ Basis: LSM (Linux Security Modules) Framework

- ▶ Anknüpfungspunkte für Sicherheitsabfragen in allen sicherheitsrelevanten Systemaufrufen
- ▶ LSM-Aufrufe erfolgen nach den "normalen" Zugriffsrecht-Abfragen (DAC)

## 8 Installation von SELinux Policies



## 6 Multi-Level-Security

G.5 Beispiel SELinux

### ■ Variante des Bell-LaPadula-Modells

### ■ Security Context wird um Sicherheitsstufe(n) $SL$ erweitert

- ▶  $SL = (s, C)$ 
  - $s$  Sensitivity, aus einer geordneten Menge von Werten
  - $C = \{c1, \dots, cn\}$  Kategorie-Werte, ungeordnete Werte
- ▶ Relationen auf Sicherheitsstufen
  - $dom$  SL1 dominates SL2  $\Leftrightarrow s1 \geq s2 \wedge C1 \supseteq C2$
  - $domby$  SL1 is dominated by SL2  $\Leftrightarrow s1 \leq s2 \wedge C1 \subseteq C2$
  - $eq$  SL1 is equal to SL2  $\Leftrightarrow s1 = s2 \wedge C1 = C2$
  - $incomp$  SL1 is incomparable to SL2  $\Leftrightarrow C1 \not\subseteq C2 \wedge C2 \not\subseteq C1$

### ■ Basis-Regeln

- ▶ Prozess darf Objekt lesen wenn  $SL_{proc} dom SL_{Obj}$
- ▶ Prozess darf Objekt schreiben wenn  $SL_{proc} domby SL_{Obj}$

### ■ Erweiterung: vertrauenswürdige Anwendungen dürfen Sicherheitsstufen von Objekten (z. B. Dateien) verändern

## 9 Feingranulare Rechte-Spezifikation bei MAC

G.5 Beispiel SELinux

### ■ MAC: Rechte werden systemweit für Objektmengen spezifiziert

- ▶ allow-Regeln  
allow Domain-Typ(en) Objekt-Typ(en) : Objekt-Klasse(n) Recht(e);

### ▲ Objekt-Klasse erlaubt Zusammenfassung/Differenzierung von Objekten

- ▶ file, blk\_file, chr\_file, dir, lnk\_file, ...
- ▶ socket, tcp\_socket, unix\_stream\_socket, ...
- ▶ msg, sem, shm
- ▶ capability, process, security, system

## 9 Feingranulare Rechte-Spezifikation bei MAC (2)

G.5 Beispiel SELinux

- ▲ Rechte für die Nutzung von Systemschnittstellen
  - im Gegensatz zu Standard-UNIX-Rechten (rwx) sehr fein-granular
- Beispiele für Objekt-Klasse *file*
  - *read*, *write*, *append*, *execute*, *ioctl*, *create*, *rename*, *unlink*,
  - *setattr* (für `chmod`), *getattr* (für `stat`),
  - *entrypoint* (s-Bit der Datei darf genutzt werden = Datei darf für Domain-Transition genutzt werden)
  - *execute\_no\_trans* (Datei wird in Domain des Aufrufers ausgeführt - ohne Domain-Transition)
  - ...
- Beispiele für Objekt-Klasse *process*
  - *fork*, *sigkill* (darf SIGKILL versenden),
  - *execstack* (Prozess das Code auf Stack ausführen)
  - ...

## 9 Feingranulare Rechte-Spezifikation bei MAC (3)

G.5 Beispiel SELinux

- ▲ Typen
  - repräsentieren Ressourcen in Bezug auf Sicherheit
  - Zuordnung erfolgt über Security Context
  - typischerweise sehr große Zahl von Typen in einem SELinux-System
- ▲ Attribute
  - Konzept zur Gruppierung von Typen
  - allow-Regeln können sich auf Attribute statt auf Typen beziehen

## 10 Literatur

MMC07. Frank Mayer, Karl MacMillan, David Caplan. *SELinux by Example*. Prentice Hall, 2007.