

Betriebssystemssicherheit

-

GNU/Linux

Björn Schießle

`<schiesbn@studi.informatik.uni-stuttgart.de>`

Universität Stuttgart

7. Februar 2006

Inhalt

- 1 Sicherheitsmodelle
 - MAC (mandatory access control)
 - RBAC (role based access control)
 - MLS (multi level security)
 - DAC (directionary access control)

Inhalt

- 1 Sicherheitsmodelle
 - MAC (mandatory access control)
 - RBAC (role based access control)
 - MLS (multi level security)
 - DAC (directionary access control)
- 2 Standardumsetzung in Linux (DAC)
 - Beispiel
 - Schwachstellen

Inhalt

- 1 Sicherheitsmodelle
 - MAC (mandatory access control)
 - RBAC (role based access control)
 - MLS (multi level security)
 - DAC (directionary access control)
- 2 Standardumsetzung in Linux (DAC)
 - Beispiel
 - Schwachstellen
- 3 SELinux
 - Entstehung
 - Umsetzung
 - Komponenten
 - Abstraktionsebenen
 - Funktionsweise

Inhalt

- 1 Sicherheitsmodelle
 - MAC (mandatory access control)
 - RBAC (role based access control)
 - MLS (multi level security)
 - DAC (directionary access control)
- 2 Standardumsetzung in Linux (DAC)
 - Beispiel
 - Schwachstellen
- 3 SELinux
 - Entstehung
 - Umsetzung
 - Komponenten
 - Abstraktionsebenen
 - Funktionsweise
- 4 Quellen

Sicherheitsmodelle - MAC

MAC (mandatory access control)

Sicherheitsmodell bei dem der Zugriff auf Objekte in Abhängigkeit von deren Markierung (Sensitivität/Label) erfolgt. Der Umfang der Rechte wird durch dynamische Regeln (Policy) bestimmt.

Kontrollmöglichkeiten

- **Administrativ definierte Policy:** Benutzer können Sicherheitsattribute nur soweit ändern wie es die Policy erlaubt.

Kontrollmöglichkeiten

- **Administrativ definierte Policy:** Benutzer können Sicherheitsattribute nur soweit ändern wie es die Policy erlaubt.
- **Kontrolle über Prozesse und Objekte:** MAC erweitert die geschützten Ressourcen weit über den üblichen Standard hinaus.

Kontrollmöglichkeiten

- **Administrativ definierte Policy:** Benutzer können Sicherheitsattribute nur soweit ändern wie es die Policy erlaubt.
- **Kontrolle über Prozesse und Objekte:** MAC erweitert die geschützten Ressourcen weit über den üblichen Standard hinaus.
- **Flexible Policy:** MAC ermöglicht die Formulierung von flexiblen Regeln, welche durch eine administrative Instanz verteilt werden. Auf diese Weise ist eine anpassungsfähige und weitreichende Kontrolle möglich.

Kontrollmöglichkeiten

- **Administrativ definierte Policy:** Benutzer können Sicherheitsattribute nur soweit ändern wie es die Policy erlaubt.
- **Kontrolle über Prozesse und Objekte:** MAC erweitert die geschützten Ressourcen weit über den üblichen Standard hinaus.
- **Flexible Policy:** MAC ermöglicht die Formulierung von flexiblen Regeln, welche durch eine administrative Instanz verteilt werden. Auf diese Weise ist eine anpassungsfähige und weitreichende Kontrolle möglich.
- **Fein abgestufte Rollenverteilung:** Root kann aufgelöst und in verschiedene Administrative Rollen zerlegt werden.

Sicherheitsmodelle - RBAC

RBAC (role based access control)

Identität wird mit einer bestimmten Rolle verbunden. Beim Anmelden an ein System wird über die Identität eine bestimmte Rolle zugewiesen welche die Zugriffsrechte beschreibt. Die Beschreibung der Zugriffsrechte basiert wiederum auf dem MAC Modell.

Sicherheitsmodelle - MLS

MLS (multi level security)

Hier werden die Objekte in Sicherheitsklassen eingestuft (z.B. vertraulich, geheim, streng-geheim). Nur wer eine ausreichende Sicherheitsstufe hat kann auf ein Objekt zugreifen.

Sicherheitsmodelle - DAC

DAC (directionary access control)

Traditionelles Sicherheitsmodell. Autorisation ist an Identität, Gruppenzugehörigkeit und Zugriffsattribute der Objekte gebunden. Entscheidung über Sicherheit liegt bei diesem Modell auch im Ermessen des Benutzers.

DAC stammt aus der Zeit in der die gemeinsame Nutzung eines Rechners und die damit verbundenen Sicherheitsrisiken im Vordergrund standen.

Standardumsetzung in Linux (DAC)

Mögliche Rechte

- *r* lesen
- *w* schreiben
- *x* ausführen
- *s* suid-Bit (set user ID) und sgid-Bit (set group ID)

Beispiel

```
- r w - r - - r - - 1 max users 6291 2005-08-02 16:16 foo.txt
```

Beispiel

```
- r w - r - - r - - 1 max users 6291 2005-08-02 16:16 foo.txt
```

- Objekt-Typ (Datei, Verzeichnis, Link,...)

Beispiel

```
- r w - r - - r - - 1 max users 6291 2005-08-02 16:16 foo.txt
```

- Objekt-Typ (Datei, Verzeichnis, Link,...)
- Rechte des Besitzers (max)

Beispiel

```
- r w - r - - r - - 1 max users 6291 2005-08-02 16:16 foo.txt
```

- Objekt-Typ (Datei, Verzeichnis, Link,...)
- Rechte des Besitzers (max)
- Rechte der Gruppe (users)

Beispiel

```
- r w - r - - r - - 1 max users 6291 2005-08-02 16:16 foo.txt
```

- Objekt-Typ (Datei, Verzeichnis, Link,...)
- Rechte des Besitzers (max)
- Rechte der Gruppe (users)
- Rechte der anderen

suid/sgid Bit

- Ohne suid/sgid Bit
 - Programm erbt Rechte von Benutzer

suid/sgid Bit

- Ohne suid/sgid Bit
 - Programm erbt Rechte von Benutzer
- Mit suid/sgid Bit
 - Programm hat Rechte des Eigentümers

Schwachstellen

- **Mensch:** Benutzer können Sicherheitsattribute von eigenen Dateien/Verzeichnissen ändern und damit anderen Objekten zugänglich machen.

Schwachstellen

- **Mensch:** Benutzer können Sicherheitsattribute von eigenen Dateien/Verzeichnissen ändern und damit anderen Objekten zugänglich machen.
- **Grobe Prozesskontrolle:** Prozesse geben Berechtigungen unbeschränkt weiter (einfache Vererbung der Berechtigung bei fork, exec). Prozesse verfügen oft über deutlich mehr Berechtigungen als zur Ausführung nötig wäre.

Schwachstellen

- **Mensch:** Benutzer können Sicherheitsattribute von eigenen Dateien/Verzeichnissen ändern und damit anderen Objekten zugänglich machen.
- **Grobe Prozesskontrolle:** Prozesse geben Berechtigungen unbeschränkt weiter (einfache Vererbung der Berechtigung bei fork, exec). Prozesse verfügen oft über deutlich mehr Berechtigungen als zur Ausführung nötig wäre.
- **Identität:** Berechtigungen sind an Identität gebunden, welche sich einfach ändern lässt (setuid, setgid)

Schwachstellen

- **Mensch:** Benutzer können Sicherheitsattribute von eigenen Dateien/Verzeichnissen ändern und damit anderen Objekten zugänglich machen.
- **Grobe Prozesskontrolle:** Prozesse geben Berechtigungen unbeschränkt weiter (einfache Vererbung der Berechtigung bei fork, exec). Prozesse verfügen oft über deutlich mehr Berechtigungen als zur Ausführung nötig wäre.
- **Identität:** Berechtigungen sind an Identität gebunden, welche sich einfach ändern lässt (setuid, setgid)
- **Grobe Rollenverteilung (Benutzer/Admin):** Es gibt keine Trennung in modulare Rollen, alle administrativen Berechtigungen werden an eine einzelne Identität gebunden (root).

SELinux

SELinux vereint alle vorgestellten Sicherheitsmodelle (DAC, MAC, MLS, RBAC) zu einem weitreichenden Sicherheitsmodell.

Entstehung

- Zwei Architektur-Prototypen für den Mach Kernel (NSA, Universität Utah, Secure Computing Corp.):
 - DT Mach (Distributed Trusted Mach)
 - DTOS (Distributed Trusted Operating System)
- Weiterentwicklung Flask(Flux Advanced Security Kernel)
- Portierung auf Linux und Lizenzänderung hin zur GPL
- Patch für Linux 2.4
- Seit Linux 2.6 offizieller Bestandteil des Kernel.

Besonderheit

Mit SELinux ist GNU/Linux das erste “Standardbetriebssystem” mit größerer Verbreitung, welches Sicherheitskonzepte verwenden, welche man sonst nur aus sicherheitskritischen Umgebungen kennt (MAC, RBAC und MLS).

Umsetzung

SELinux ist ein “Aufsatz” auf das Standard-Sicherheitsmodell (DAC).

- Beide Zugriffskontrollen müssen Zugriff erlauben: DAC und SELinux(RBAC, MLS, MAC)
- Zuerst werden die Rechte des Dateisystems (DAC) überprüft, dann die anderen (SELinux)

Komponenten

Security Server

Der Security Server ist die zentrale Komponente der SELinux Architektur und trifft alle sicherheitsrelevanten Entscheidungen.

Komponenten

Security Server

Der Security Server ist die zentrale Komponente der SELinux Architektur und trifft alle sicherheitsrelevanten Entscheidungen.

Objektmanager

Der Objektmanager übernimmt die Verwaltung der Sicherheitsattribute, sorgt für die korrekte Zuordnung zu den Objekten (Dateien, Prozesse, Sockets,..) und setzt letztlich die Entscheidungen des Security Servers um.

Objekt

Alle Komponenten auf die Zugriffen wird (Dateien, Verzeichnisse, Sockets,..).

Objekt

Alle Komponenten auf die Zugriffen wird (Dateien, Verzeichnisse, Sockets,..).

Subjekt

Handelnde Komponenten im System (Prozesse).

Security-Contexts

Die Security-Contexts sind die Basis für die Entscheidungen des Security-Server.

Ein Security-Context fasst verschiedene Sicherheitsattribute zusammen:

- Benutzeridentität
- Rolle des Benutzers
- Typ des Objekts/Prozess

Dabei ist zu beachten, dass nur dem Security-Server bekannte Kombinationen möglich sind.

Benutzer

- Der Begriff “Benutzer” bezieht sich bei SELinux in der Regel auf reale Personen, die Zugang zum System haben (Ausnahme: `system_u`). Zusätzliche Pseudo-User für bestimmte Prozesse sind aber nichtmehr nötig, da die Rechte von Prozessen über den “Typ” festgelegt werden.
- Die Benutzerverwaltung von SELinux ist unabhängig von der User-ID in GNU/Linux.

Vorteile:

- SELinux-Benutzeridentität lässt sich nach dem Login nichtmehr verändern. Wechsel der Berechtigung ist nur noch über die “Rolle” oder des “Typs” möglich.
- Mehrere Benutzer lassen zu einem unprivilegierten Benutzer zusammenfassen (`user_u`), dadurch bleibt die Komplexität der Regeln überschaubarer.

Rolle

Je nach Aufgabe treten die Benutzer in verschiedenen Rollen auf. Die Rechte werden dann über die jeweilige Rolle zugeteilt.

Beispiel:

- Systemprozesse: system_r
- Administratoren: sysadmin_r
- normale Benutzer: user_r

Typ

Jedem Objekt/Prozess wird ein Typ zugeordnet. Ein Objekt/Prozess kann einen eigenen Typ haben oder diesen mit anderen Objekten/Prozessen teilen.

Gelegentlich wird zwischen Typ und Domain unterschieden. Der Typ beschränkt sich dann auf Objekte (Dateien,...) und die Domain bezieht sich auf Subjekte (Prozesse,...)

Typ

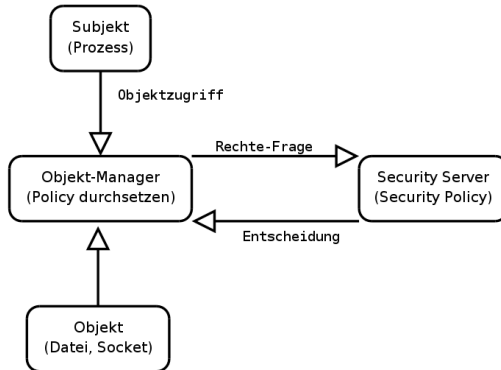
Jedem Objekt/Prozess wird ein Typ zugeordnet. Ein Objekt/Prozess kann einen eigenen Typ haben oder diesen mit anderen Objekten/Prozessen teilen.

Gelegentlich wird zwischen Typ und Domain unterschieden. Der Typ beschränkt sich dann auf Objekte (Dateien,...) und die Domain bezieht sich auf Subjekte (Prozesse,...)

Type Enforcement (TE)

Type Enforcement bezeichnet die in SELinux umgesetzte MAC Variante. TE sorgt dafür, dass Beziehungen zwischen Typen eingehalten werden.

Entscheidungsfindung



Access Vector Cache (AVC)

Im Gegensatz zum normalen Linuxkernel fragt der Objekt-Manager bei SELinux bei jedem Schreib- und Lesezugriff ab, ob die Aktion noch erlaubt ist.



Quellen

- www.nsa.gov/selinux/
- Regel-recht - Security Enhanced Linux im Einsatz, Carsten Grohmann, Linux Magazin 01/2003
(http://www.carstengrohmann.de/download/publications/Artikel_SELinux_LinuxMagazin_2003_01.pdf)
- <http://www.burde-consulting.de/pdf/selinux.pdf>

Some Rights Reserved



This work is licensed under the Creative Commons
Attribution-ShareAlike License.

To view a copy of this license, visit

<http://creativecommons.org/licenses/by-sa/2.0/de/deed.de>

or send a letter to

Creative Commons, 559 Nathan Abbott Way, Stanford, California
94305, USA.