Mandatory Access Control durch SELinux

9. April 2005

H. Görl
goerl@luga.de

Überblick

1. Sicherheit allgemein
2. Defizite unter Linux/Unix
3. LSM + SELinux

Sicherheit — wozu?

- Kostet Geld und Zeit
- Alles ist sowieso unsicher!
- Angriffe und Bedrohungen eh unklar!
- (Meist) keine kriminellen Handlungen, also egal
- Andere sind schuld!

Aber:

- Informationen werden wichtiger

- Angriffe werden gefährlicher
- Datenschutz
- Informationsschutz
- Schutz der Persönlichkeit

(Unsichtbare) Gefahren

1. Identitätsraub
2. Diebstahl und Weitergabe
3. Enferntes zerstören
4. Automatisierte Angriffe

Standard Linux Sicherheitsmechanismen

- ACLs (`rwx`–Rechtevergabe) und eingeschränkte Capabilities
- Virtuelle Speicherverwaltung zum Speicherschutz
- Modulare Authentifizierungsmechanismen (PAM)
- Schnelle Fehlerbehebung
- Open Source
- Loopback Verschlüsselung

(Erhebliche) Defizite

1. `root` Benutzer ist allmächtig (Bsp.)
2. Rechtevergabe i.W. für Dateien und Verzeichnisse
3. Rechteprüfung ist statisch (Bsp.)
4. `/proc` Dateisystem nur schlecht schützbar
5. Wirkungsvolle Mechanismen gegen schlampige Konzepte (Bsp.)

NSA SELinux – Entwicklung

- Entwickelt von NSA und SCC (Secure Computing Corporation)
- Rahmenarchitektur Flask
- Eigentlich eine flexible MAC–Architektur (Mandatory Access Control)
- Policy–Modell kann ausgetauscht werden
- Trennung vom Regelwerk und der Durchsetzung
- Prototypen waren DTMach und DTOS (Mikrokerne)
- Formale Methoden wurden für das Design angewendet
- Zusammen mit Uni. of Utah Integration in Flux / Fluke

NSA SELinux – Totale Überwachung?

Die Aussage

> „For those of you looking for a paranoia level of security for your Linux box, let the NSA help you secure your system.“

hat irgendwie ähnliche Züge wie

> Sicherheitsdialog „Microsoft immer vertrauen? JA!“

- Aber: SELinux ist Open Source
- Aber: Konzepte sind klar
- Aber: Keine „Security by Obscurity“

SELinux – Überblick

- Im Jahr 2000 Start der NSA mit SELinux (kein Mikrokern)
- MAC für Verwaltung von
  1. Prozessen
  2. Dateien
  3. Sockets
- Zunächst nur grundlegendes Rahmenwerk für spezielle Policies
- NAI Labs. haben dann konkrete Security Manager und Policies realisiert

SELinux

- Security Policy Logik ist im Security Server realisiert
- Strikte Trennung zwischen Policy und Durchsetzung
- Policy Schnittstelle genau definiert
- Objekt Manager zur Durchsetzung der Regeln
- Access Vector Cache (AVC)
- Security Server Beispiel besitzt folgende Mechanismen
  - Role Based Access Control
  - Multi Level Security (MLS)

  - Type Enforcement (TE)

- Label sollen auch stellenübergreifend verwendet werden
- DTOS verwendet dazu IPSec
- In SELinux soll es mit FreeSWAN realisiert werden

SELinux – Begriffe

- **identity:** feste Beschreibung der Identität
- **domain:** Sicherheitsrahmen für Subjekte, z.B. Prozesse
- **type:** Sicherheitsbeschreibung für Objekte, z.B. Dateien
- **role:** Regel, die Umgebungen beschreiben, z.B.:

  `role user_r types user_passwd_t`

- **policy:** Menge von Rollen
- **security context:** `user:role:type`

SELinux – Kernel I

```
[*] Enable different security models
[*]   Socket and Networking Security Hooks
<M>   Default Linux Capabilities
<M>   Root Plug Support
<M>   BSD Secure Levels
[*]   NSA SELinux Support
[*]     NSA SELinux boot parameter
(1)      NSA SELinux boot parameter default value (NEW)
[*]     NSA SELinux runtime disable
[*]     NSA SELinux Development Support
[*]     NSA SELinux AVC Statistics
[*]     NSA SELinux MLS policy (EXPERIMENTAL)
```

SELinux – Kernel II

```
<*> Ext3 journalling file system support
[*]   Ext3 extended attributes
[*]     Ext3 POSIX Access Control Lists
[*]     Ext3 Security Labels
```

SELinux – Pakete

- Neben Kernel noch Usermodule und Biblitoheken nötig:
  - policycoreutils
  - selinux-policy-default
  - checkpolicy
  - libselinux1
  - selinux-doc
  - selinux-utils

- Standardkomponenten/Pakete werden ersetzt:
  - bsdutils
  - coreutils
  - cron
  - fileutils
  - findutils
  - libpam
  - login
  - logrotate
  - mount
  - passwd
  - procps
  - psmisc
  - shellutils
  - ssh
  - textutils
  - util-linux

SELinux – Konfiguration

- `selinux` Mountpoint erstellen:

  /etc/fstab: `none /selinux selinuxfs defaults 0 0`

- Dateisystem labeln: `make relabel`
- PAM-Modul einbinden
- und dann booten und beten!

Nach dem ersten Reboot:

- Warnungen ignorieren

- Debian-Tools mit Prefix verwenden!
- Normaler Benutzer hat Kontext

  `<user>:user_r:user_t`
- root hat Kontext

  `root:sysadm_r:sysadm_t`
- Wechsel der Rolle mit `newrole -r sysadm_r`

SELinux – Einsatz

- Zuerst Unix Rechte, dann SELinux Rechte:
- Datei `/etc/shadow`

  $-rw-r-----$ root shadow system_u:object_r:shadow_t /etc/shadow
- Programm `/usr/bin/passwd`

  $-rwsr-xr-x$ root root system_u:object_r:passwd_exec_t /usr/bin/passwd
- ginge eigentlich NICHT! Aber:
- Type-Wandlung durch Rule

  type_transition userdomain passwd_exec_t:process passwd_t;
- Rule erlauben

1. allow shadow_t passwd_exec_t:file entrypoint;
2. allow userdomain passwd_t:process transition;

- Und genau den Zugriff erlauben:

  allow passwd_t shadow_t:file create ioctl read ...;

SELinux – Einsatz

Weitere Projekte

- RSBAC (MAC, Compartments, MLS wie SysV)
- LIDS – Linux Intrusion Detection System
- Medusa DS9, Kern Zugriffskontrollen
- TrustedBSD, MAC, Starre Politiken
- LOMAC – Low Water–Mark Integrity Protection (Integrität)
- Kryptomechanismen als Kernerweiterung (Kernadressraum)
- Kryptomechanismen im Benutzeradressraum (OpenSSL, crypto++, cryptlib, cryptix, usw.)

- Erweiterung der PAM–Module, etwa für Chipkartenauthentifikation
- Abgesicherte Dateisysteme
- Security Audits
- „Trusted Computing“ (TCPA, NGSCB, Nexus)

Literatur und Quellen

- Zwingend: `www.nsa.gov/selinux/`
- Bill McCarty, „SELINUX – NSA's Open Source Security Enhanced Linux“, O'Reilly Verlag
- SELinux Distribution: `http://selinux.sourceforge.net/`
- SELinux inofficial FAQ:
  `http://sourceforge.net/docman/display_doc.php?docid=14882&group_id=21266`
- SELinux official FAQ: `www.nsa.gov/selinux/info/faq.cfm`