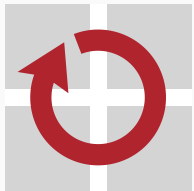
AppArmor

Praktikum angewandte Systemsoftwaretechnik

11.11.2019

Michael Kupfer, Kay Friedrich

Friedrich-Alexander-Universität Erlangen-Nürnberg



Lehrstuhl für Verteilte Systeme
und Betriebssysteme

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Motivation

Motivation

- Standardmäßig: Discretionary Access Control (DAC)
  - Ressourcen werden Benutzern und Gruppen zugeordnet
  - Prozesse sind zur Laufzeit einem Benutzer zugeordnet
  - Zugriffe auf Ressourcen werden augrund des Benutzers erlaubt oder abgelehnt

Motivation

- Standardmäßig: Discretionary Access Control (DAC)
  - Ressourcen werden Benutzern und Gruppen zugeordnet
  - Prozesse sind zur Laufzeit einem Benutzer zugeordnet
  - Zugriffe auf Ressourcen werden augrund des Benutzers erlaubt oder abgelehnt
- Sehr grobe Zugriffsbeschränkung
  - Prozesse haben mehr Rechte als sie tatsächlich benötigen
  - Kann für Sicherheitslücken/Rechteausweitung genutzt werden

Motivation

- Standardmäßig: Discretionary Access Control (DAC)
  - Ressourcen werden Benutzern und Gruppen zugeordnet
  - Prozesse sind zur Laufzeit einem Benutzer zugeordnet
  - Zugriffe auf Ressourcen werden augrund des Benutzers erlaubt oder abgelehnt

- Sehr grobe Zugriffsbeschränkung
  - Prozesse haben mehr Rechte als sie tatsächlich benötigen
  - Kann für Sicherheitslücken/Rechteausweitung genutzt werden

→ AppArmor:
  - Mandatory Access Control (MAC)
  - Feingranularere Vergabe von Rechten

Entwicklungsgeschichte

Entwicklungsgeschichte

- 90er Jahre: Vorgänger SubDomain

Entwicklungsgeschichte

- 90er Jahre: Vorgänger SubDomain
- Nicht das einzige Projekt, das diesen Ansatz verfolgt
  - Kein Projekt soll im Kernel bevorzugt werden
  - Entwicklung des Linux Security Module (LSM)
    $\rightarrow$ bietet einheitliche Schnittstelle

Entwicklungsgeschichte

- 90er Jahre: Vorgänger SubDomain
- Nicht das einzige Projekt, das diesen Ansatz verfolgt
  - Kein Projekt soll im Kernel bevorzugt werden
  - Entwicklung des Linux Security Module (LSM)
    $\rightarrow$ bietet einheitliche Schnittstelle
- SubDomain wird zur Nutzung von LSM umgeschrieben

Entwicklungsgeschichte

- 90er Jahre: Vorgänger SubDomain
- Nicht das einzige Projekt, das diesen Ansatz verfolgt
  - Kein Projekt soll im Kernel bevorzugt werden
  - Entwicklung des Linux Security Module (LSM)
    $\rightarrow$ bietet einheitliche Schnittstelle
- SubDomain wird zur Nutzung von LSM umgeschrieben
- 2005: Umbenennung in AppArmor und Aufnahme des Kernel-Moduls in den Upstream Kernel

Entwicklungsgeschichte

- 90er Jahre: Vorgänger SubDomain
- Nicht das einzige Projekt, das diesen Ansatz verfolgt
  - Kein Projekt soll im Kernel bevorzugt werden
  - Entwicklung des Linux Security Module (LSM)
    $\rightarrow$ bietet einheitliche Schnittstelle
- SubDomain wird zur Nutzung von LSM umgeschrieben
- 2005: Umbenennung in AppArmor und Aufnahme des Kernel-Moduls in den Upstream Kernel
- 2009: Canonical übernimmt Entwicklung von AppArmor

Aufbau & Funktionsweise

Kernel-Konfiguration

LSM- und AppArmor-Support

- CONFIG_SECURITY=y
- CONFIG_AUDIT=y
- CONFIG_SECURITY_APPARMOR=y

Kernel-Konfiguration

LSM- und AppArmor-Support

- CONFIG_SECURITY=y
- CONFIG_AUDIT=y
- CONFIG_SECURITY_APPARMOR=y

AppArmor-Kernel-Modul aktivieren

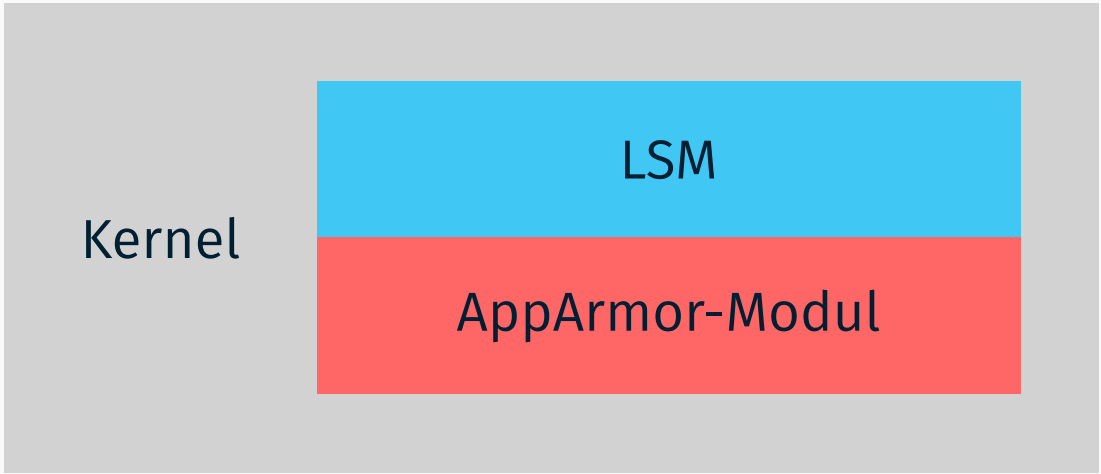
- in der Kernel-Konfiguration
  - CONFIG_SECURITY_APPARMOR_BOOTPARAM_VALUE=1
  - CONFIG_DEFAULT_SECURITY_APPARMOR=y
- oder per Kernel-Parameter
  - apparmor=1
  - security=apparmor

Linux Security Module

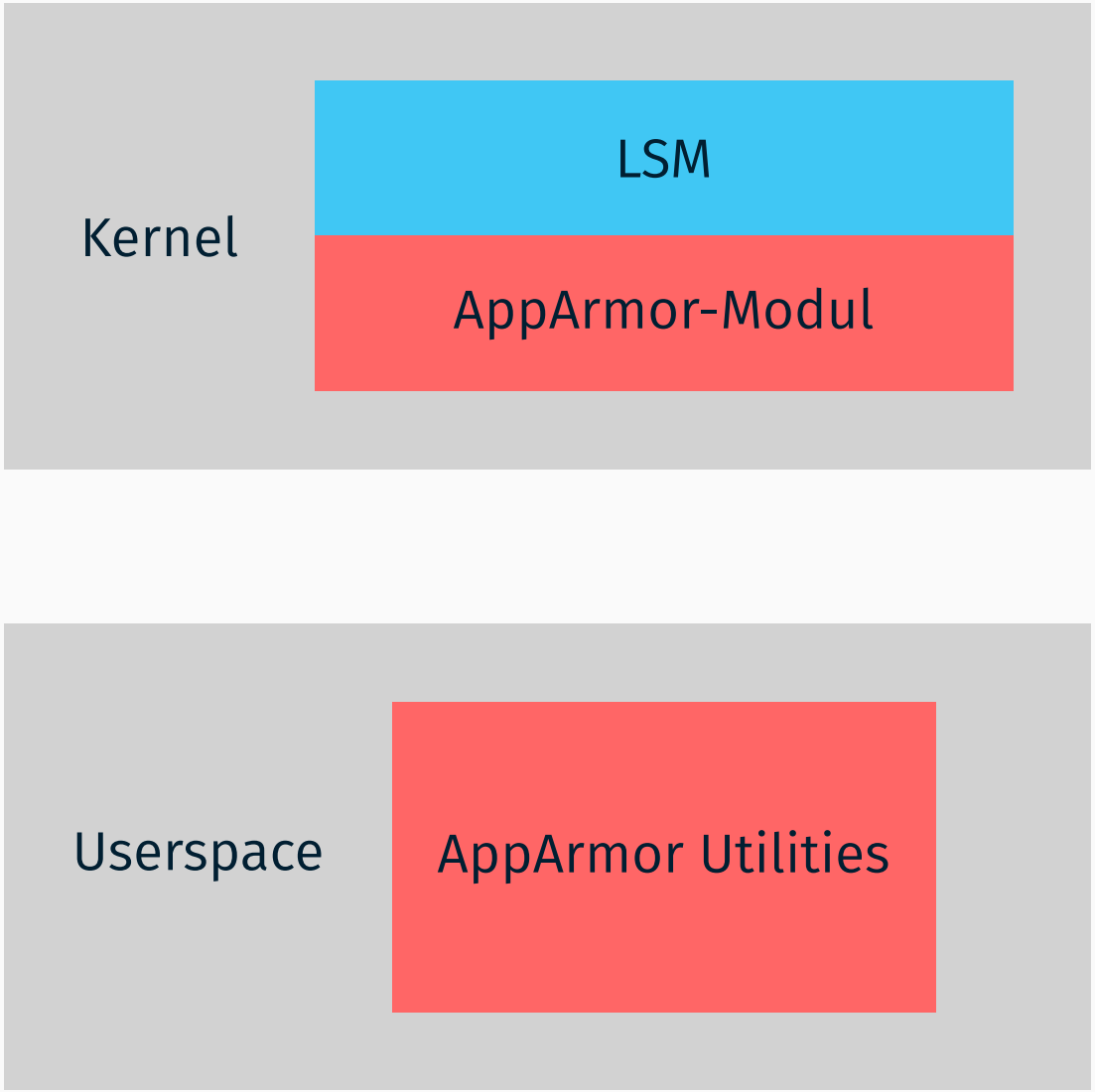
AppArmor muss sich bei LSM registrieren:

```
static struct security_operations apparmor_ops = {
    .name =                     "apparmor",
                            ...
    .capget =                   apparmor_capget,
    .capable =                  apparmor_capable,
                            ...
    .path_mkdir =               apparmor_path_mkdir,
    .path_rmdir =               apparmor_path_rmdir,
                            ...
    .file_open =                apparmor_file_open,
                            ...
};
```

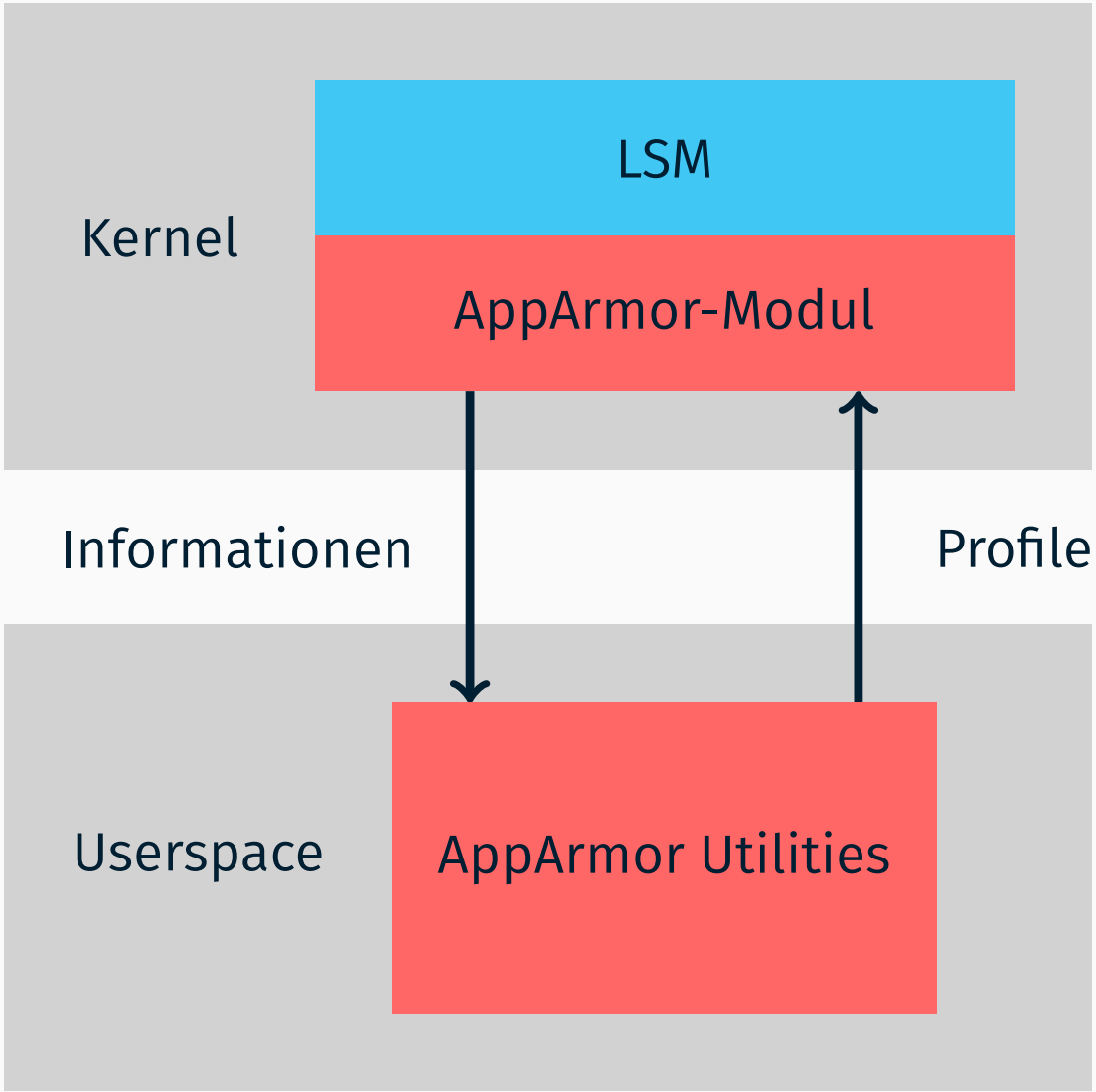
Funktionsweise



Funktionsweise



Funktionsweise



Anwendung

AppArmor Policies

Policies

Die AppArmor Policy besteht aus den Profilen in **/etc/apparmor.d**. Diese werden kompiliert und in den Kernel geladen.

Einsatzbereich der wichtigsten Userspace-Tools:

- **aa-genprof**: Anlegen eines neuen Profils (interaktiv)
- **aa-logprof**: Editieren von Profil (interaktiv)
- **aa-enforce**: Programm wechselt in Enforce-Mode
- **aa-complain**: Programm wechselt in Complain-Mode
- **apparmor_parser**: Kompilieren nach manuellem Editieren

Profiles

```
#include <file>
profile usr.bin.example {  # comment 1
    # comment 2
    /home/*/foo rw,
    /usr/bin/example_child ix, #inherit profile
    deny /**.py x, #deny execution of python files
}
```

Berechtigungen

- **r**: **R**ead
- **w**: **W**rite
- **x**: E**X**ecute
- **m**: **M**emory map executable (Bibliothek laden)
- **l**: **L**ink (verlinke Files)
- **k**: Loc**K**

Profiles

Weitere Regeln

- **capability**: Zugriff auf capabilities zulassen (siehe `man capabilities`).
  `capability setuid`
- **network**: Erlaube Netzwerkzugriff.
  `network inet dgram`
- **deny**: Explizit Zugriff verbieten.
  `deny /path/to/file w`
- **audit**: Logge den Zugriff auf die Datei.
  `audit /path/to/file2 w`

Beispiel Zugriff auf Nutzerdateien

```
@{HOME}/ r,
@{HOME}/** r,
owner @{HOME}/** w,

audit deny @{HOME}/.ssh/** mrwkl,
audit deny @{HOME}/.gnome2_private/** mrwkl,
audit deny @{HOME}/.kde{,4}/share/apps/kwallet/** mrwkl,

# Allow read to all files user has DAC access to
# and write for files the user
# owns on removable media and filesystems.
/media/** r,
/mnt/** r,
/srv/** r,
/net/** r,
owner /media/** w,
owner /mnt/** w,
owner /srv/** w,
owner /net/** w,
```

Infrastruktur

Code-Management

- Gitlab mit:
  - Kernel-Code
  - Userspace-Code
  - Profilen
  - Wiki
  - Dokumentation mit Tutorials
- Bugreporting über:
  - Launchpad
  - `apparmor@lists.ubuntu.com` für öffentliche Bugs
  - `security@apparmor.net` für sicherheitsrelevante Bugs

Organisation

Organisation innerhalb des Projekts

Members

- 12 Entwickler in Gitlab
- John Johansen und Steve Beattie sind derzeitig ’Projekt-Owner’
- Monatliches Treffen im IRC-Channel

Zum Projekt Beitragen

- Userspace-Tools:
  - Patches über die Mailingliste oder als Merge Request
  - Zustimmung von mind. 1 weiteren Commit-Berechtigten
  - Bei Verweigerung des Patches sollten Dikussionen angestoßen und neue Patches eingesendet werden
- Kernel-Modul: Standard Kernel Commit-Policy

Fragen?

Quellen (1)

Apparmor lsm.c.
`https://git.kernel.org/pub/scm/linux/kernel/git/jmorris/linux-security.git/tree/security/apparmor/lsm.c`.
06.11.2019.

Dokumentation.
`https://gitlab.com/apparmor/apparmor/-/wiki_pages/Documentation`.
04.11.2019.

Kernel dokumentation.
`https://gitlab.com/apparmor/apparmor/-/wiki_pages/TechnicalDoc_Kernel`.
04.11.2019.

Quellen (2)

- Launchpad.
  https://launchpad.net/apparmor.
  04.11.2019.
- Release prozess.
  https://gitlab.com/apparmor/apparmor/-/wiki_pages/ReleaseProcess.
  04.11.2019.
- Wiki.
  https://gitlab.com/apparmor/apparmor/-/wiki_pages/home.
  04.11.2019.