

# LINUX ADVANCED SECURITY

FAST-TRACK

SVA



# AGENDA

- > [Motivation](#)
- > [Grundlagen](#)
- > [SELinux](#)
- > [AppArmor](#)
- > [OpenVAS](#)
- > [Dev-Sec](#)
- > [fail2ban](#)

# AUFGABENÜBERSICHT

- > [Lab 01: Systeme analysieren/aufräumen und automatische Updates aktivieren](#)
- > [Lab 02: Dateisystem-Berechtigungen überprüfen und anpassen](#)
- > [Lab 03: ServerTokens anpassen](#)
- > [Lab 04: AIDE verwenden](#)
- > [Lab 05: SELinux entdecken](#)
- > [Lab 06: SELinux-Dateikontext überprüfen](#)
- > [Lab 07: SELinux-Booleans](#)
- > [Lab 08: SELinux-Ports verwalten](#)
- > [Lab 09: AppArmor entdecken](#)

# AUFGABENÜBERSICHT

- > [Lab 10: AppArmor-Profil erstellen](#)
- > [Lab 11: AppArmor verwalten](#)
- > [Lab 12: OpenVAS Host-Audit](#)
- > [Lab 13: fail2ban einsetzen](#)
- > [Lab 14: Anwenden von Dev-Sec](#)
- > [Lab 15: Erstellen eines InSpec-Profiles](#)

# // MOTIVATION

# MOTIVATION

- > Linux-Systeme werden im Server-Bereich immer **populärer\***
  - > 80% der Webserver werden unter Linux betrieben
  - > mehr als 50% der Azure-Workloads sind Linux-Workloads
  - > **IoT**-Technologie ist i.d.R. Linux-basiert

# MOTIVATION

- > Linux-Systeme werden im Server-Bereich immer **populärer\***
  - > 80% der Webserver werden unter Linux betrieben
  - > mehr als 50% der Azure-Workloads sind Linux-Workloads
  - > **IoT**-Technologie ist i.d.R. Linux-basiert
- > Dadurch steigt das Interesse für Angreifende
- > Standard-Installationen sind leider i.d.R. **nicht sicher**
  - > **Trade-off**: Sicherheit oder Komfort

\* das Jahr des Linux-Desktops kommt sicher noch

# MOTIVATION

- > Glücklicherweise gibt es zahlreiche Möglichkeiten Linux zu härten, u.a.:
  - > Sinnvolle Grundeinstellungen
  - > **SELinux** und AppArmor
  - > Automatisierte **Härtung** nach Best Practices
  - > Automatisches Sperren von IPs
- > Ausgiebiges Testen ist **unabdingbar**

# // GRUNDLAGEN

# SECURITY 101

## PAKETAUSWAHL

- > *weniger ist mehr*
  - > minimale Paketauswahl
  - > keine **GUI** auf Servern installieren
  - > nicht benötigte Pakete deinstallieren

# SECURITY 101

## PAKETAUSWAHL

- > *weniger ist mehr*
  - > minimale Paketauswahl
  - > keine **GUI** auf Servern installieren
  - > nicht benötigte Pakete deinstallieren
- > keine Pakete aus nicht **vertrauenswürdigen** Quellen
- > Regelmäßig **Updates** installieren
  - > `apt unattended-upgrades`
  - > `yum-cron`

# SECURITY 101

## FIREWALL

- > Firewall **nicht** deaktivieren
- > sinnhaftige Konfiguration vornehmen
- > Zonen-Konfiguration überprüfen

# SECURITY 101

## FIREWALL

- > Firewall **nicht** deaktivieren
- > sinnhaftige Konfiguration vornehmen
- > Zonen-Konfiguration überprüfen

## SELINUX

- > **Nicht** deaktivieren
- > Im Fehlerfall in **Permissive**-Modus versetzen und Fehler untersuchen

# SECURITY 101

## DATEISYSTEM

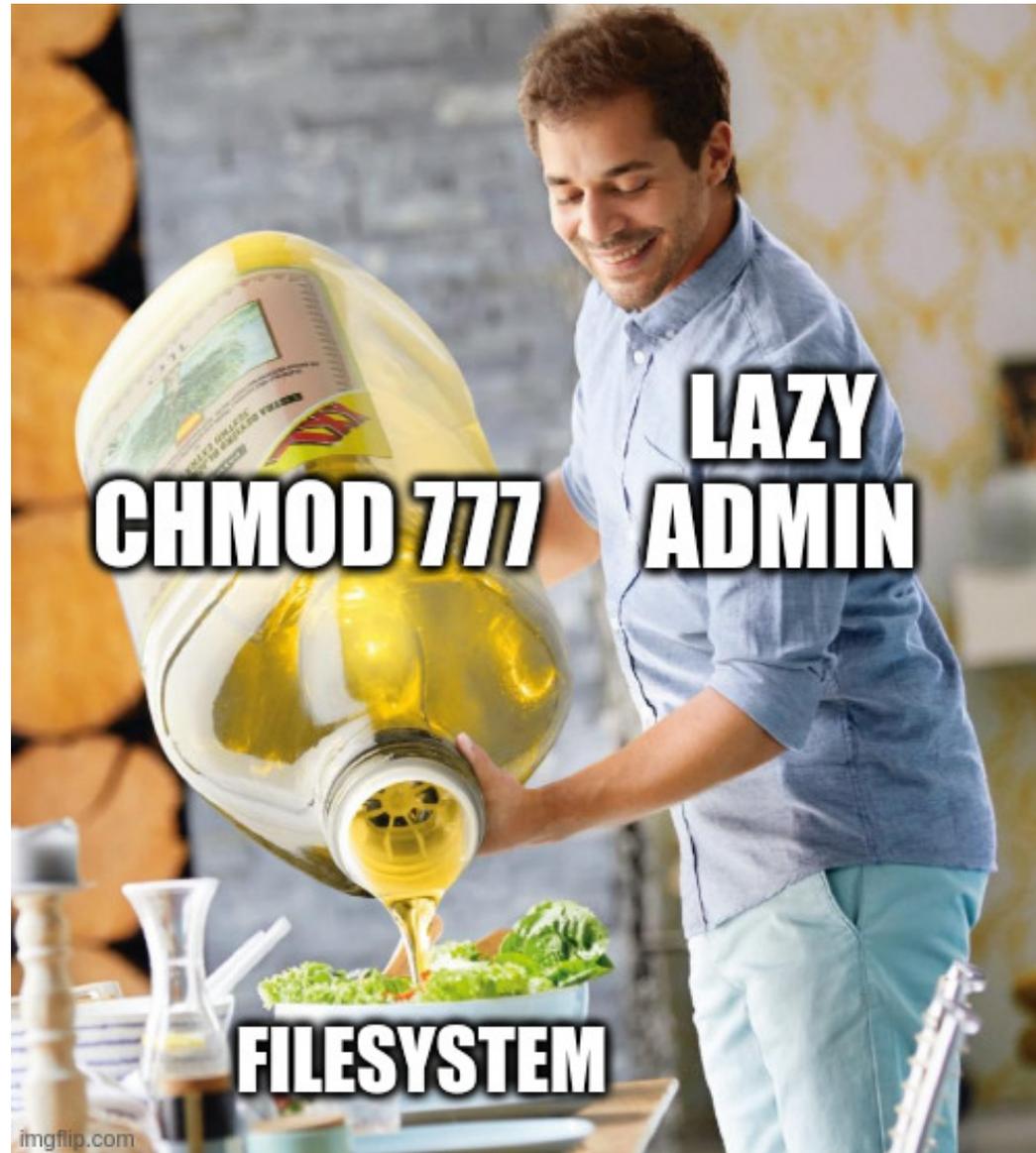
- > niemals `chmod 0777` definieren
- > **ACLs** einsetzen wo klassische Berechtigungen nicht ausreichen
- > Einsatz von **setuid** und **setgid** vermeiden

# SECURITY 101

## DATEISYSTEM

- > niemals `chmod 0777` definieren
- > **ACLs** einsetzen wo klassische Berechtigungen nicht ausreichen
- > Einsatz von **setuid** und **setgid** vermeiden
- > Dedizierte Partitionen einsetzen (*LVM oder Btrfs*)\*
  - > `/var`
  - > `/home`
  - > `/opt`

\* verhindert Fehler durch erschöpften Speicherplatz



imgflip.com

# WIEDERHOLUNG: SETUID, SETGID, STICKY BIT

Bit	Darstellung	Datei	Verzeichnis
setuid	s / 4	Ausführung mit <code>owner</code> -Berechtigungen	-
setgid	s / 2	Ausführung mit <code>group</code> -Berechtigungen	Neue Dateien erhalten Gruppe des Elternverzeichnisses
sticky	t / 1	-	Nur Datei-/Verzeichnis- <code>owner</code> können löschen

- > Das `setuid`-Bit findet vor allem dort Verwendung, wo **unprivilegierte** User administrative Tasks ausführen können müssen
  - > Beispiel: `passwd`: Ändern von `/etc/shadow`
- > Das `sticky`-Bit findet vor allem bei geteilten Verzeichnissen Verwendung
  - > Beispiel: `/tmp`, nicht jede:r soll Dateien anderer User löschen können
- > Die Anzahl an Dateien mit zusätzlichen Bits ist überschaubar und sollte gelegentlich **überprüft** werden
  - > wird gerne als **Einfallstor** genutzt

# WIEDERHOLUNG: SETUID, SETGID, STICKY BIT

Suchen von Dateien/Verzeichnissen mit zu großzügigen Berechtigungen:

```
# find / -type f -perm 777
```

Suchen von Dateien/Verzeichnissen mit setuid-, setgid- und sticky-Bits:

```
# find / -perm /4000  
# find / -perm /2000  
# find / -perm /1000
```



# SECURITY 101

## PASSWÖRTER

- > **Aging** und **Länge** konfigurieren (`/etc/login.defs`)
  - > `PASS_MIN_DAYS`, `PASS_MAX_DAYS`
  - > `PASS_MIN_LEN`, `PASS_MAX_LEN`
- > **Komplexität** konfigurieren (`/etc/security/pwquality.conf`)
- > LDAPS statt LDAP einsetzen

# SECURITY 101

## SSH

- > Root-Zugang deaktivieren (`PermitRootLogin no`)
- > alte Chiffren deaktivieren
- > TCP-Forwarding/-Tunneling deaktivieren (`AllowTcpForwarding no`,  
`PermitTunnel no`)

# SECURITY 101

## SSH

- > Root-Zugang deaktivieren (`PermitRootLogin no`)
- > alte Chiffren deaktivieren
- > TCP-Forwarding/-Tunneling deaktivieren (`AllowTcpForwarding no`,  
`PermitTunnel no`)
- > Public Key-Authentifizierung aktivieren (`PubkeyAuthentication yes`)
- > Passwort-Authentifizierung deaktivieren (`PasswordAuthentication no`)
- > Leere Passwörter deaktivieren (`PermitEmptyPasswords no`)

# SECURITY 101

## SERVERDIENSTE

- > OS-/Anwendungsversionen verstecken (*Banner*, *ServerTokens*)
- > Regelmäßige Backups konfigurieren und **Restores** testen
- > **Testsysteme** bereitstellen

# SECURITY 101

## SERVERDIENSTE

- > OS-/Anwendungsversionen verstecken (*Banner*, *ServerTokens*)
- > Regelmäßige Backups konfigurieren und **Restores** testen
- > **Testsysteme** bereitstellen

## DMZ

- > VPN-Zwang für SSH
- > Honeypots einsetzen
- > Automatisches Sperren von bruteforcenden IPs

# EXKURS: EXPLOIT-DATENBANKEN

- > Server-Dienste geben i.d.R. **mehr** Informationen preis **als notwendig**
  - > z.B. benutzte Linux-Distribution und Server-Software
  - > **Debug**-Ausgaben von Web-Anwendungen
- > In öffentlichen Umgebungen ergibt es Sinn, dieses Verhalten **abzustellen**

# EXKURS: EXPLOIT-DATENBANKEN

- > Server-Dienste geben i.d.R. **mehr** Informationen preis **als notwendig**
  - > z.B. benutzte Linux-Distribution und Server-Software
  - > **Debug**-Ausgaben von Web-Anwendungen
- > In öffentlichen Umgebungen ergibt es Sinn, dieses Verhalten **abzustellen**
- > Im Internet kursieren mehrere Datenbanken mit funktionalen **Exploits**:
  - > [Offensive Security Exploit Database](#)
  - > [Rapid7 Vulnerability and Exploit Database](#)
  - > [VulDB](#) - dokumentiert **seit 1970** Verwundbarkeiten
  - > [CXSecurity](#).

# LAB 01

SYSTEME ANALYSIEREN/AUFRÄUMEN UND AUTOMATISCHE  
UPDATES AKTIVIEREN

# LAB 02

## DATEISYSTEM-BERECHTIGUNGEN ÜBERPRÜFEN UND ANPASSEN

# LAB 03

## SERVERTOKENS ANPASSEN

# AIDE

- > **A**dvanced **I**ntrusion **D**etection **E**nvironment
- > überprüft die **I**ntegrität von Dateien und Verzeichnissen
- > für **R**ootkit-Analyse geeignet

# AIDE

- > **A**dvanced **I**ntrusion **D**etection **E**nvironment
- > überprüft die **I**ntegrität von Dateien und Verzeichnissen
- > für **R**ootkit-Analyse geeignet
- > erstellt Datenbanken mit kryptografischen **P**rüfsummen
  - > unterstützt u.a. **M**D5, **S**HA-1/256/512 und **C**RC32
  - > kann mit POSIX ACLs, SELinux und `xattrs` umgehen
- > Konfigurationsdatei definiert, welche Dateien/Verzeichnisse überprüft werden

# AIDE

- > **A**dvanced **I**ntrusion **D**etection **E**nvironment
- > überprüft die **I**ntegrität von Dateien und Verzeichnissen
- > für **R**ootkit-Analyse geeignet
- > erstellt Datenbanken mit kryptografischen **P**rüfsummen
  - > unterstützt u.a. **M**D5, **S**HA-1/256/512 und **C**RC32
  - > kann mit POSIX ACLs, SELinux und `xattrs` umgehen
- > Konfigurationsdatei definiert, welche Dateien/Verzeichnisse überprüft werden
- > regelmäßige Ausführung stellt Integrität sicher
- > **A**bweichungen werden aufgelistet, Mail-Report möglich

# AIDE

- > Die Datenbank befindet sich unter `/var/lib/aide/aide.db.gz`
  - > beim Vergleich wird eine neue Datenbank `aide.db.new.gz` erstellt
- > Protokollausgaben finden sich unter `/var/log/aide/aide.log`

# AIDE

- > Die Datenbank befindet sich unter `/var/lib/aide/aide.db.gz`
  - > beim Vergleich wird eine neue Datenbank `aide.db.new.gz` erstellt
- > Protokollausgaben finden sich unter `/var/log/aide/aide.log`
- > Das Regelwerk unterstützt zahlreiche **Flags**
  - > Berechtigungen, Datei-Änderungen, Benutzer/Gruppe, Größe,...
- > **Templates** erleichtern die Definition neuer Regeln
  - > Verzeichnisse, Inhaltliche Änderungen,...

# FLAGS (AUSSCHNITT)

Flag	p	i	u	g	s	m / a
Bedeutung	Berechtigungen	inodes	Benutzer	Gruppe	Größe	Zeitstempel Anpassung/Zugriff

Flag	acl	selinux	xattrs	md5 / sha1 / sha256 / ...
Bedeutung	ACLs	SELinux-Kontext	Erweiterte Datei-Attribute	Prüfsumme

# TEMPLATES (AUSSCHNITT)

```
NORMAL = p+i+n+u+g+s+m+c+acl+selinux+xattrs+sha512
# For directories, don't bother doing hashes
DIR = p+i+n+u+g+acl+selinux+xattrs
# Access control only
PERMS = p+u+g+acl+selinux+xattrs
# Logfile are special, in that they often change
LOG = p+u+g+n+S+acl+selinux+xattrs
# Content + file type.
CONTENT = sha512+ftype
# Extended content + file type + access.
CONTENT_EX = sha512+ftype+p+u+g+n+acl+selinux+xattrs
```

# BEISPIELE

```
/boot          CONTENT_EX
/opt           CONTENT

# These are too volatile
!/usr/src
!/usr/tmp

# Pkg manager
/etc/dnf CONTENT_EX
/etc/yum.conf$ CONTENT_EX
/etc/yum CONTENT_EX
/etc/yum.repos.d CONTENT_EX

# web server
/etc/httpd CONTENT_EX
```

# LAB 04

## AIDE VERWENDEN

# ZUSAMMENFASSUNG

- > eine **minimale** Paketauswahl aus **vertrauenswürdiger** Quelle ist sinnvoll
- > automatische Updates
- > Sicherheitsmechanismen (*Firewall, SELinux*) **nicht deaktivieren**
- > Dateisystem-Berechtigungen regelmäßig **überprüfen**
- > sinnhaftige Passwort-Richtlinien festlegen

# ZUSAMMENFASSUNG

- > eine **minimale** Paketauswahl aus **vertrauenswürdiger** Quelle ist sinnvoll
- > automatische Updates
- > Sicherheitsmechanismen (*Firewall, SELinux*) **nicht deaktivieren**
- > Dateisystem-Berechtigungen regelmäßig **überprüfen**
- > sinnhaftige Passwort-Richtlinien festlegen
- > SSH-Zugriff **erschweren**
  - > kein Root-Zugriff
  - > Schlüssel- statt Kennwort-Authentifizierung
- > Rootkit-Analyse durch Einsatz von **AIDE** o.ä. erschweren

// SELINUX

# LINUX KERNEL SECURITY MODULES

- > Framework im Linux-Kernel, um erweiterte Sicherheitsmodule einzubinden
  - > **AppArmor** (1998)
  - > **SELinux** (2000)
  - > **SMACK** (2008)
  - > **Tomoyo Linux** (2009)

# LINUX KERNEL SECURITY MODULES

- > Framework im Linux-Kernel, um erweiterte Sicherheitsmodule einzubinden
  - > **AppArmor** (1998)
  - > **SELinux** (2000)
  - > **SMACK** (2008)
  - > Tomoyo Linux (2009)
- > Kernel löst Sicherheitsmodul durch **Hooks** in Kernel-Modulen aus
  - > z.B. vorherige Authentisierung nach Anfragen einer Datei
- > ist jedoch nicht für **Logging** zuständig
  - > von Sicherheitsmodulen implementiert

# WAS IST SELINUX?

- > **SE**curity Enhanced **Linux**
- > Kernel-Erweiterung, welche **Mandatory Access Control** implementiert
  - > Bestandteil des Kernels seit 2.6.x
  - > In Android seit 4.3 enthalten
- > wird vor allem von Red Hat und der NSA entwickelt\*

# WAS IST SELINUX?

- > **SE**curity Enhanced **Linux**
- > Kernel-Erweiterung, welche **Mandatory Access Control** implementiert
  - > Bestandteil des Kernels seit 2.6.x
  - > In Android seit 4.3 enthalten
- > wird vor allem von Red Hat und der NSA entwickelt\*
- > Unter Fedora und RHEL vorinstalliert
- > Optional auch unter Debian, Ubuntu und openSUSE/SLES verfügbar
  - > erfahrungsgemäß sind hier vordefinierte Regeln **weniger umfangreich**

\* jedoch selbstverständlich Open-Source

# EXKURS: MANDATORY ACCESS CONTROL

- > in den **1970**ern erstmals spezifiziert
- > dient der Sicherheit von unauthorisiertem Zugriff (**Vertraulichkeit**)
- > verhindert Umgehung auf Anwendungs-/Kernelebene (**Integrität**)

# EXKURS: MANDATORY ACCESS CONTROL

- > in den **1970**ern erstmals spezifiziert
- > dient der Sicherheit von unauthorisiertem Zugriff (**Vertraulichkeit**)
- > verhindert Umgehung auf Anwendungs-/Kernelebene (**Integrität**)
- > definiert, **welche** Ressourcen Programme und Dienste sehen dürfen
  - > *so wenig wie möglich, so viel wie nötig*
  - > erschwert Schaden durch Fehlkonfiguration und Ausnutzen von Lücken
  - > erreicht durch **ergänzende Regeln** und Eigenschaften

# EXKURS: MANDATORY ACCESS CONTROL

- > in den **1970**ern erstmals spezifiziert
- > dient der Sicherheit von unauthorisiertem Zugriff (**Vertraulichkeit**)
- > verhindert Umgehung auf Anwendungs-/Kernelebene (**Integrität**)
- > definiert, **welche** Ressourcen Programme und Dienste sehen dürfen
  - > *so wenig wie möglich, so viel wie nötig*
  - > erschwert Schaden durch Fehlkonfiguration und Ausnutzen von Lücken
  - > erreicht durch **ergänzende Regeln** und Eigenschaften
- > **Multi-Level Security** ergänzt um **Schutzklassen** (*Bell-La Padula-Modell*)
  - > Streng geheim, Geheim, Vertraulich, Öffentlich
  - > optional, nicht standardmäßig aktiviert

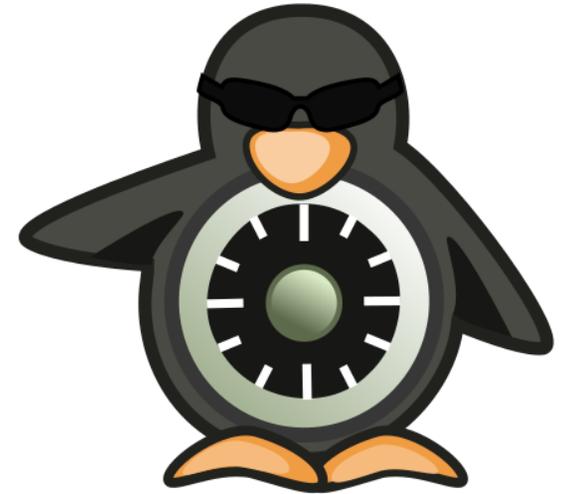
# WO GREIFT SELINUX EIN?

SELinux **erkennt** und steuert **Zugriffe**:

- > Prozesse
- > Netzwerk-Ports
- > Dateisystem-Zugriffe (*über zusätzliche Flags*)

Was SELinux **nicht** ist/sein sollte:

- > Antiviren-Software
- > Ersatz für Firewalls und sichere Passwörter
- > eine einzige Security-Lösung, die alles andere ersetzt



# WO GREIFT SELINUX EIN?

SELinux klärt die folgende Frage:

Darf **Subjekt** eine **Aktion** mit **Objekt** durchführen?

# WO GREIFT SELINUX EIN?

SELinux klärt die folgende Frage:

Darf **Subjekt** eine **Aktion** mit **Objekt** durchführen?

## BEISPIELE

*Darf **Webserver-User** **Dateien** von User `ppinkepank` **lesen**?*

# WO GREIFT SELINUX EIN?

SELinux klärt die folgende Frage:

Darf **Subjekt** eine **Aktion** mit **Objekt** durchführen?

## BEISPIELE

*Darf **Webserver-User** **Dateien** von *User ppinkepank* **lesen**?* - Nein 🙅

# WO GREIFT SELINUX EIN?

SELinux klärt die folgende Frage:

Darf **Subjekt** eine **Aktion** mit **Objekt** durchführen?

## BEISPIELE

Darf *Webserver-User* *Dateien von User ppinkepank* **lesen**? - Nein 🙅

Darf *MySQL-User* *Dateien unterhalb /var/lib/mysql* **schreiben**?

# WO GREIFT SELINUX EIN?

SELinux klärt die folgende Frage:

Darf **Subjekt** eine **Aktion** mit **Objekt** durchführen?

## BEISPIELE

Darf *Webserver-User* *Dateien von User ppinkepank* **lesen**? - Nein 🙅

Darf *MySQL-User* *Dateien unterhalb /var/lib/mysql* **schreiben**? - Ja 👍

# SELINUX 101

- > Prozesse, Netzwerk-Ports, User, Dateien und Ordner haben ein **Security Label**
  - > auch **SELinux-Kontext** oder **-Label** genannt
- > Label wird in zahlreichen **SELinux-Policies** referenziert, um Zugriffe zu regeln
  - > i.d.R. Policy pro Dienst oder Anwendung

# SELINUX 101

- > Prozesse, Netzwerk-Ports, User, Dateien und Ordner haben ein **Security Label**
  - > auch **SELinux-Kontext** oder **-Label** genannt
- > Label wird in zahlreichen **SELinux-Policies** referenziert, um Zugriffe zu regeln
  - > i.d.R. Policy pro Dienst oder Anwendung
- > SELinux verbietet alle Zugriffe, die nicht **explizit erlaubt** sind
  - > großer Unterschied gegenüber AppArmor
- > SELinux setzt **nach** klassischen UNIX-Berechtigungen ein
  - > keine Verstöße bei Dateien, die man ohnehin nicht benutzen darf
  - > Effekt: trotz `chmod 777` dürfen Dateien nicht benutzt werden

# SELINUX 101

SELinux unterstützt drei **Modi** und **Typen**:

Modus	Erklärung
enforcing	aktiv ( <i>Standard</i> )
permissive	aktiv, Regelverstöße werden nicht unterbunden; <b>Troubleshooting</b>
disabled	inaktiv ( <i>nicht benutzen!</i> )
Typ	Erklärung
targeted	alle Prozesse werden reglementiert ( <i>Standard</i> )
minimum	nur ausgewählte Prozesse werden reglementiert
mls	<b>Multi-Level Security</b>

Seriously, stop disabling SELinux.  
[Learn how to use it](#) before you shut it off.

Every time you run `setenforce 0`, you make [Dan Walsh](#) weep.  
Dan is a nice guy and he certainly doesn't deserve that.



Quellen: [stopdisablinglinux.com](https://stopdisablinglinux.com) / [people.redhat.com](https://people.redhat.com)

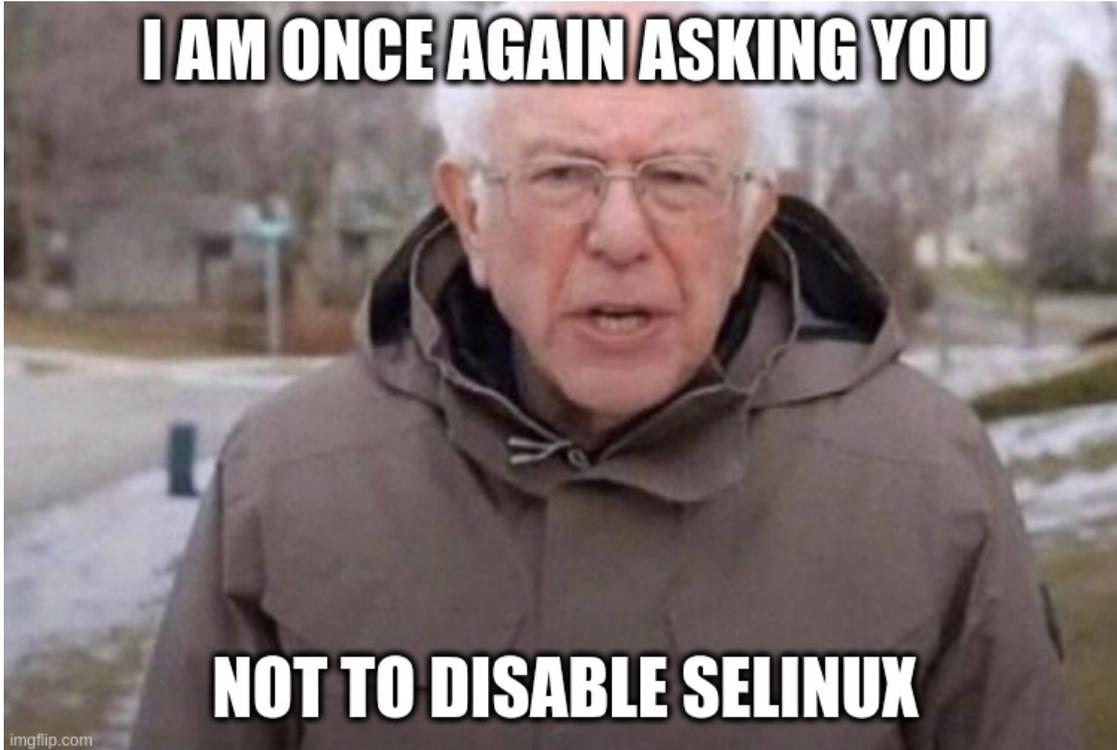
# SELINUX BITTE NICHT DEAKTIVIEREN

- > SELinux ist eine **sinnvolle** Maßnahme
  - > lernen damit umzugehen, statt es abzuschalten
- > korrekt angewendet, verursacht SELinux nur noch selten Probleme
  - > inzwischen bedeutend besser von Third-Party-Software unterstützt

# SELINUX BITTE NICHT DEAKTIVIEREN

- > SELinux ist eine **sinnvolle** Maßnahme
  - > lernen damit umzugehen, statt es abzuschalten
- > korrekt angewendet, verursacht SELinux nur noch selten Probleme
  - > inzwischen bedeutend besser von Third-Party-Software unterstützt
- > wenn erforderlich, **Permissive** statt **Disabled** wählen
- > Beim Wechsel von **Disabled** nach **Enforcing**, sollte das Dateisystem **re-labeled** werden
  - > ansonsten kommt es zu zahlreichen Fehlermeldungen
  - > dauert je nach Größe lange

```
# touch /.autorelabel  
# systemctl reboot
```



# SELINUX 101

Ein **SELinux-Kontext** besteht aus verschiedenen **Feldern**:

- > `user` - SELinux-User
- > `role` - dazugehörige **RBAC\***-Rolle

\* **Role-Based Access Control**

# SELINUX 101

Ein **SELinux-Kontext** besteht aus verschiedenen **Feldern**:

- > `user` - SELinux-User
- > `role` - dazugehörige **RBAC\***-Rolle
- > **type**
  - > das wichtigste Feld, endet i.d.R. auf `_t`
- > `security_level`
  - > für **MLS** relevant, ansonsten immer `s0`

\* **Role-Based Access Control**

# SELINUX 101

- > einige Tools haben **spezielle Parameter**, um mit SELinux umzugehen
  - > z.B.: `-Z` und `--context` bei `ls` und `ps`

```
# ls -lZd /var/www/html /var/lib/mysql
drwxr-xr-x. 4 mysql mysql system_u:object_r:mysqld_db_t:s0      4096 Mar 12 14
drwxr-xr-x. 2 root  root  system_u:object_r:httpd_sys_content_t:s0  116 Mar  8 16
```

- > Der SELinux-User (`system_u`) unterscheidet sich zu den UNIX-Usern (`mysql` bzw. `root`)
- > die dazugehörige SELinux-Rolle lautet `object_r`
- > Die **Typen** unterscheiden nach Applikation: `mysql_d_db_t`, `httpd_sys_content_t`

# LAB 05

## SELINUX ENTDECKEN

# USER UND ROLLEN

- > SELinux pflegt eine eigene **Userdatenbank**
- > User müssen keinen Zusammenhang zu Linux-Usern haben
- > User sind über **Rollen** einer **Policy** zugeordnet
  - > mehrere Rollen pro User möglich
  - > diese Policy definiert Rechte und Einschränkungen (**Type Enforcement**)

# USER UND ROLLEN

- > SELinux pflegt eine eigene **Userdatenbank**
- > User müssen keinen Zusammenhang zu Linux-Usern haben
- > User sind über **Rollen** einer **Policy** zugeordnet
  - > mehrere Rollen pro User möglich
  - > diese Policy definiert Rechte und Einschränkungen (**Type Enforcement**)
- > vom User ausgeführte Prozesse\* **vererben** diese Definitionen
- > definiert bei MLS auch, in welchen **Leveln** interagiert werden kann

\* auch **Domains** genannt

# USER UND ROLLEN

Anzeigen definierter User:

```
# semanage user -l
```

<b>SELinux</b> User	Labeling Prefix	<b>MLS/</b> MCS Level	<b>MLS/</b> MCS Range	<b>SELinux</b> Roles
guest_u	user	s0	s0	guest_r
root	user	s0	s0-s0:c0.c1023	staff_r sysadm_r system_r u
staff_u	user	s0	s0-s0:c0.c1023	staff_r sysadm_r system_r u

Ändern der Rollen eines Users:

```
# semanage user -m -R "unconfined_r staff_r" staff_u
```

# DATEI-KONTEXTE

SELinux-Kontext kann über `ls -Z` eingesehen werden:

```
# ls -lZd /var/www/html
drwxr-xr-x. 2 root root system_u:object_r:httpd_sys_content_t:s0 116 Mar  8 16:05
```

- > der Ordner `/var/www/html` gehört dem SELinux-**User** `system_u`
- > den **Typ** `httpd_sys_content_t` ermöglicht dem Apache-Dienst darauf zuzugreifen

# DATEI-KONTEXTE

SELinux-Kontext kann über `ls -Z` eingesehen werden:

```
# ls -lZd /var/www/html
drwxr-xr-x. 2 root root system_u:object_r:httpd_sys_content_t:s0 116 Mar  8 16:05
```

- > der Ordner `/var/www/html` gehört dem SELinux-**User** `system_u`
- > den **Typ** `httpd_sys_content_t` ermöglicht dem Apache-Dienst darauf zuzugreifen

Der Kontext kann **temporär** über das `chcon`-Kommando geändert werden:

```
# chcon -t user_home_dir_t -R /var/www/html
# ls -lZd /var/www/html
drwxr-xr-x. 2 root root system_u:object_r:user_home_dir_t:s0 134 Mar 13 09:30 /va
```

# DATEI-KONTEXTE

In diesem Fall kann der Webserver **nicht mehr** auf die Dateien **zugreifen**:

```
$ curl -s -o /dev/null -w "%{http_code}" http://localhost
403
```

Der HTTP-Code **403** (*Unauthorized*) signalisiert die fehlenden Berechtigungen.

Temporäre Änderungen können mit `restorecon` rückgängig gemacht werden:

```
# restorecon -Rv /var/www/html
Relabeled /var/www/html from system_u:object_r:user_home_dir_t:s0 to
system_u:object_r:httpd_sys_content_t:s0
Relabeled /var/www/html/index.html from unconfined_u:object_r:user_home_t:s0 to
unconfined_u:object_r:httpd_sys_content_t:s0
...
```

# DATEI-KONTEXTE

- > temporäre Änderungen sollten nach Tests **immer persistiert** werden
- > der Befehl `semanage` definiert u.a. **Standard-Labels** pro Pfad
  - > nach Definition eines Musters wird dieses mit `restorecon` angewendet

# DATEI-KONTEXTE

- > temporäre Änderungen sollten nach Tests **immer persistiert** werden
- > der Befehl `semanage` definiert u.a. **Standard-Labels** pro Pfad
  - > nach Definition eines Musters wird dieses mit `restorecon` angewendet
- > notwendig, wenn Dienste **abweichende Pfade** nutzen sollen
  - > z.B. MySQL-Datenbank unterhalb `/data/mysql`

```
# semanage fcontext -a -t mysqld_db_t "/data/mysql(/.*)?"  
# restorecon -Rv /data/mysql  
# systemctl start mariadb.service
```



imgflip.com

# LAB 06

## SELINUX-DATEIKONTEXT ÜBERPRÜFEN

# LOGGING

Fehler werden in das systemweite Audit\* geschrieben:

```
# grep denied /var/log/audit/audit.log
type=AVC msg=audit(1710322253.511:458): avc: denied { read } for pid=837
comm="httpd" name="index.html" dev="dm-0" ino=402849944 tclass=file permissive=0
scontext=system_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:user_home_t:s0
```

ausearch formatiert den Regelverstoß etwas besser:

```
# ausearch -m AVC -ts boot
----
time->Wed Mar 13 09:30:53 2024
type=PROCTITLE msg=audit(1710322253.511:458): proctitle=2F7573722F7362696E2F6874
type=SYSCALL msg=audit(1710322253.511:458): arch=c000003e syscall=257 success=no
type=AVC msg=audit(1710322253.511:458): avc: denied { read } for pid=837 comm=
```

\* /var/log/audit/audit.log

# TROUBLESHOOTING

audit2why hilft dabei, verständliche Fehlermeldungen anzuzeigen:

```
# audit2why -i /var/log/audit/audit.log
...
type=AVC msg=audit(1710322253.511:458): avc: denied { read } for pid=837 ...
```

Was caused **by**:

The **boolean** httpd\_read\_user\_content was **set** incorrectly.

Description:

Allow httpd **to read** user content

Allow access **by** executing:

```
# setsebool -P httpd_read_user_content 1
```

Hier wird auch gleich eine potentielle Lösung angeboten:

> Aktivieren eines **Booleans**

# TROUBLESHOOTING

- > manchmal sind die Fehlermeldungen weniger hilfreich
- > in diesem Fall kann das Blocken von Regelverstößen **temporär** deaktiviert werden\*
  - > auch **permissive** Mode genannt
  - > für einzelne oder alle Dienste

\* es ist **immer** die Firewall, DNS oder SELinux 🙄

# TROUBLESHOOTING

- > manchmal sind die Fehlermeldungen weniger hilfreich
- > in diesem Fall kann das Blocken von Regelverstößen **temporär** deaktiviert werden\*
  - > auch **permissive** Mode genannt
  - > für einzelne oder alle Dienste

Kommando	Beschreibung
<code>setenforce 0/1</code>	Deaktiviert/Aktiviert Blocken von Regelverstößen temporär
<code>semanage permissive -a httpd_t</code>	Webserver-Prozesse <b>permissive</b> schalten
<code>semanage permissive -d httpd_t</code>	Webserver-Prozesse wieder <b>enforcing</b> schalten
<code>semanage permissive -l</code>	Zeigt alle permissive Domains

\* es ist **immer** die Firewall, DNS oder SELinux 🤪

# TROUBLESHOOTING: ÜBLICHES VORGEHEN

Überprüfen, ob die betroffene Anwendung im **Permissive Mode** läuft:

```
# setenforce 0
systemctl restart shittyapp.service
```

Falls ja, **Enforcing Mode** re-aktivieren und Logs clearen:

```
# setenforce 1
# > /var/log/audit/audit.log
```

Anwendung neustarten und Regelverstöße **analysieren**:

```
# systemctl restart service
# audit2why -i /var/log/audit/audit.log
```

# BOOLEANS

- > **steuern** der **Verhalten** von SELinux-Policies
  - > aktivieren/deaktivieren einzelne Regeln
- > werden i.d.R. verwendet um **einzelne Ausnahmen** zu definieren
  - > z.B. Webserver Zugriff auf Userhomes ermöglichen

# BOOLEANS

- > **steuern** der **Verhalten** von SELinux-Policies
  - > aktivieren/deaktivieren einzelne Regeln
- > werden i.d.R. verwendet um **einzelne Ausnahmen** zu definieren
  - > z.B. Webserver Zugriff auf Userhomes ermöglichen

Kommando	Beschreibung
<code>getsebool [-a]</code>	Einzelne/Alle Booleans anzeigen
<code>setsebool [-P] NAME WERT</code>	Boolean temporär/dauerhaft setzen

# BOOLEANS

## Anzeigen aller Booleans:

```
# getsebool -a  
abrt_anon_write --> off  
abrt_handle_event --> off  
...
```

## Dauerhaftes Setzen eines Booleans:

```
# setsebool -P httpd_read_user_content 1
```

# LAB 07

## SELINUX-BOOLEANS

# PORTS

- > auch **Netzwerkports** werden mit Labels versehen
  - > z.B. `ssh_port_t` für **tcp/22**
- > so kann verhindert werden, dass Anwendungen auf **unüblichen Ports** lauschen
  - > z.B. um **Schadecode** nach einem Angriff nachzuladen

# PORTS

- > auch **Netzwerkports** werden mit Labels versehen
  - > z.B. `ssh_port_t` für **tcp/22**
- > so kann verhindert werden, dass Anwendungen auf **unüblichen Ports** lauschen
  - > z.B. um **Schadecode** nach einem Angriff nachzuladen
- > Port-Definitionen können über `semanage` verwaltet werden
  - > notwendig, wenn Dienste auf Nicht-Standard-Ports lauschen sollen

Befehl	Beschreibung
<code>semanage port -l</code>	Auflisten von Port-Definitionen
<code>semanage port -a -t TYP -p tcp/udp PORT</code>	Hinzufügen einer Definition
<code>semanage port -d -t TYP -p tcp/udp PORT</code>	Entfernen einer Definition
<code>semanage port -D -t TYP</code>	Entfernen aller Definitionen

# LAB 08

## SELINUX-PORTS VERWALTEN

# DOKUMENTATION

Jedes SELinux-*(Sub)*Kommando hat eine **Manpage**:

```
$ man semanage
$ man semanage-fcontext
$ man semanage-port
```

Die **einzelnen** Applikationen verfügen auch über Manpages\*:

```
$ man apache_selinux
$ man mysqld_selinux
$ man zabbix_selinux
```

\* Teil des `selinux-policy-doc`-Pakets

# POLICY-MODULE

- > Sammlung verschiedener SELinux-**Regeln**
- > werden **kompiliert** und in den **Kernel** geladen
  - > dieser regelt den Zugriff dann auf höchstener Ebene
- > werden vorkompiliert ausgeliefert

# POLICY-MODULE

- > Sammlung verschiedener SELinux-**Regeln**
- > werden **kompiliert** und in den **Kernel** geladen
  - > dieser regelt den Zugriff dann auf höchstener Ebene
- > werden vorkompiliert ausgeliefert
- > haben eine **Priorität** zwischen **1** und **999**
  - > Standard: **100**
  - > bei gleichem Namen wird das Modul mit der **höheren** Priorität geladen

# POLICY-MODULE VERWALTEN

Befehl	Beschreibung
<code>semodule -l</code>	Auflisten installierter Policy-Module
<code>semodule -lfull</code>	Auflisten aller Policies mit Priorität
<code>semodule -i NAME.pp</code>	Installieren eines Policy-Moduls
<code>semodule -i NAME.pp -X 200*</code>	Modul mit angepasster Priorität laden
<code>semodule -r NAME</code>	Entfernen eines Moduls
<code>semodule -e NAME</code>	Modul aktivieren
<code>semodule -d NAME</code>	Modul deaktivieren

\* Langform: `--priority`

# POLICY-MODUL ERSTELLEN

## > kein triviales Thema

- > nur selten benötigt, i.d.R. bei der Software-Entwicklung
- > für mitgelieferte Software werden Policies ausgeliefert
- > erfordert **tiefes Verständnis** der Applikation und SELinux

## > Vorgehen

- > Erstellen von Modul-Quellcode mittels `sepolicy generate`
- > Modul mit `make` erstellen
- > Modul mit `semodule` in den Kernel laden



# ZUSAMMENFASSUNG

- > **Linux Kernel Security Modules** ermöglichen den Einsatz weiterer Sicherheitsmodule
  - > u.a. AppArmor und SELinux
- > SELinux ergänzt Linux um **Mandatory Access Control**
  - > definiert welche Ressourcen von Programmen und Dienst genutzt werden dürfen
  - > unterbindet Verstöße auf **höchster Ebene**
  - > alles, was nicht **explizit erlaubt** ist, ist verboten

# ZUSAMMENFASSUNG

- > **Linux Kernel Security Modules** ermöglichen den Einsatz weiterer Sicherheitsmodule
  - > u.a. AppArmor und SELinux
- > SELinux ergänzt Linux um **Mandatory Access Control**
  - > definiert welche Ressourcen von Programmen und Dienst genutzt werden dürfen
  - > unterbindet Verstöße auf **höchster Ebene**
  - > alles, was nicht **explizit erlaubt** ist, ist verboten
- > **Multi-Level Security** kann optional um **Schutzklassen** ergänzen
  - > z.B. Streng geheim, Geheim, Vertraulich, Öffentlich
- > SELinux ist vor allen auf **Red Hat**-artigen Distributionen beliebt
  - > erkennt und steuert Zugriffe auf Prozesse, Netzwerk-Ports und das Dateisystem
  - > wird vor allem durch Ressourcen-Kennzeichnung (**Kontext**) erreicht

# ZUSAMMENFASSUNG

- > Ein **Kontext** besteht aus einem **User**, einer **Rolle** und einem **Typ**
  - > es existiert eine dedizierte User-Datenbank
  - > Rollen sprechen Usern Berechtigungen zu
  - > Typen verbinden Ressourcen mit Prozessen (**Domains**)
- > SELinux-**Policies** beinhalten Zugriffs-/Verbotsregeln
  - > **Booleans** können das Verhalten beeinflussen

# ZUSAMMENFASSUNG

- > Ein **Kontext** besteht aus einem **User**, einer **Rolle** und einem **Typ**
  - > es existiert eine dedizierte User-Datenbank
  - > Rollen sprechen Usern Berechtigungen zu
  - > Typen verbinden Ressourcen mit Prozessen (**Domains**)
- > SELinux-**Policies** beinhalten Zugriffs-/Verbotsregeln
  - > **Booleans** können das Verhalten beeinflussen
- > Verschiedene Tools können beim **Troubleshooting** helfen
  - > `ausearch`
  - > `setenforce`
  - > `audit2why`
- > SELinux und AppArmor sind nur **Teilkomponenten** eines sicheren Systems
  - > sie ersetzen weder Firewalls noch Antiviren-Software

# // APPARMOR

# APPARMOR

- > **1998** erstmalig vorgestellt
- > eines der anerkannten Linux Kernel Security Modules
  - > steuert Zugriffsrechte einzelner Prozesse auf **Systemebene**

# APPARMOR

- > **1998** erstmalig vorgestellt
- > eines der anerkannten Linux Kernel Security Modules
  - > steuert Zugriffsrechte einzelner Prozesse auf **Systemebene**
- > bietet **Access Control** u.a. für:
  - > Dateien
  - > Capabilities
  - > Netzwerk
  - > Mount, Remount, Umount
  - > DBUS
  - > Unix Sockets

# PROFILE

- > Definition von Berechtigungen einer Anwendung
- > Anwendung darf nur, was das Profil zulässt
- > Liegen unterhalb `/etc/apparmor.d`

# MODI

- > `enforce` (*Zwangsmodus*) - Aktionen mit Regelverstößen werden dokumentiert und verboten
- > `complain` (*Lernmodus*) - Aktionen mit Regelverstößen werden dokumentiert, aber nicht unterbunden
- > `audit` (*Prüfmodus*) - Alle Regelanwendungen und -verstöße werden geloggt

# KOMMANDOS

Kommando	Beschreibung
<code>aa-enabled</code>	Gibt an, ob AppArmor aktiv ist
<code>aa-status</code>	Status-Überblick mit Profilen und Modi
<code>aa-unconfined</code>	Liste von Prozessen mit ungeschütztem Netzzugriff
<code>aa-audit</code>	Profil in Audit-Modus versetzen
<code>aa-complain</code>	Profil in Complain-Modus versetzen
<code>aa-enforce</code>	Profil in Enforce-Modus versetzen
<code>aa-autodep</code>	Profil im Complain-Modus erstellen
<code>aa-logprof</code>	Profil aufgrund von Meldungen in <code>/var/log/syslog</code> ergänzen
<code>aa-cleanprof</code>	Aufräumen eines Profils
<code>aa-easyprof</code>	Erstellt ein leeres Profil
<code>apparmor_parser</code>	Lädt/entfernt Profil in den/aus dem Kernel
<code>aa-teardown</code>	Entfernt alle geladenen Profile aus dem Kernel

# APPARMOR-STATUS ÜBERPRÜFEN

```
# aa-enabled
Yes

# aa-status
apparmor module is loaded.
44 profiles are loaded.
24 profiles are in enforce mode.
    /usr/bin/man
    ...
20 profiles are in complain mode.
    /usr/bin/irssi
    ...
1 processes have profiles defined.
1 processes are in enforce mode.
    /usr/sbin/haveged (552)
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
```

# LOGGING

```
$ grep apparmor /var/log/syslog
```

Profil deaktivieren:

```
ln -s /etc/apparmor.d/<profil> /etc/apparmor.d/disable/<profil>
```

# LAB 09

## APPARMOR ENTDECKEN

# ORDNERSTRUKTUR

- > `/etc/apparmor.d` - AppArmor-Profile, Tuneables und Abstractions
- > `/etc/apparmor.d/tunables/*` - Sammlung von Variablen, die für Profile verwendet werden (z.B. Home-Verzeichnisse)
- > `/etc/apparmor.d/abstractions/*` - Vordefinierte Snippets für Anwendungen

# TUNABLES UND ABSTRACTIONS

- > **Tunables** = Variablen, die für Profile verwendet werden, z.B.
  - > HOME und HOMEDIRS, enthalten Pfade zu Heimatverzeichnissen
  - > `system_share_dirs / user_share_dirs`, enthalten Pfade zu share-Verzeichnissen

# TUNABLES UND ABSTRACTIONS

- > **Tunables** = Variablen, die für Profile verwendet werden, z.B.
  - > HOME und HOMEDIRS, enthalten Pfade zu Heimatverzeichnissen
  - > `system_share_dirs / user_share_dirs`, enthalten Pfade zu share-Verzeichnissen
- > **Abstractions** = Sammlung nützlicher Variablen für Profile
- > sollen das Erstellen eigener Profile **vereinfachen**, z.B.
  - > `python` - Pfade üblicher Python-Anwendungen
  - > `gnome` - Pfade für GTK-Anwendungen

# TUNABLES UND ABSTRACTIONS

## Tunables für Heimatverzeichnisse:

```
# @{HOME} is a space-separated list of all user home directories. While
# it doesn't refer to a specific home directory (AppArmor doesn't
# enforce discretionary access controls) it can be used as if it did
# refer to a specific home directory
@{HOME}=@{HOMEDIRS}/* /root/

# @{HOMEDIRS} is a space-separated list of where user home directories
# are stored, for programs that must enumerate all home directories on a
# system.
@{HOMEDIRS}=/home/
...
```

Quelle: `/etc/apparmor.d/tunables/home`

# TUNABLES UND ABSTRACTIONS

## Abstractions für Python-Anwendungen:

```
# Site-wide configuration
/etc/python{2.[4-7],3.[0-9]}/** r,

# shared python paths
/usr/share/{pyshared,pycentral,python-support}/** r,
/{var,usr}/lib/{pyshared,pycentral,python-support}/** r,
/usr/lib/{pyshared,pycentral,python-support}/** .so mr,
/var/lib/{pyshared,pycentral,python-support}/** .pyc mr,
/usr/lib/python3/dist-packages/** .so mr,
...
```

Quelle: `/etc/apparmor.d/abstractions/python`

# BEISPIELPROFIL

```
#include <tunables/global>

/usr/local/bin/myapp {
  #include <abstractions/base>

  capability sys_admin,
  deny network,

  owner /home/*/.bashrc r,
  /usr/bin/env ix,
  /usr/bin/python3.8 mrx,
  /usr/local/bin/myapp r,
}
```

- > `#include` - Importieren von **Tunables** (*Variables*) oder **Abstractions** (*vordefinierte Regeln*)
- > `capability` - grundlegende [Eigenschaften/Merkmale](#)
- > `network` - Netzwerk-Regelungen
- > Dateipfade mit/ohne Wildcards und ergänzender Flags

# FLAGS

- > Berechtigungen

- > `r` - lesen

- > `w` - schreiben

- > `m` - `mmap`-Funktion nutzen

- > `x` - ausführen

- > Flags

- > `audit` - alle Regelanwendungen werden protokolliert

- > `owner` - Zugriff nur gestattet, falls Datei dem ausführenden User gehört

- > `deny` - Zugriff wird explizit verboten

# CAPABILITIES

- > Capabilities sind Teil des Linux-Kernels, siehe auch `man 7 capabilities`
  - > `CAP_SYS_ADMIN` wird zu `sys_admin`

Capability	Beschreibung
<code>chown, setgid, setuid</code>	UIDs/GIDs von Dateien und Prozessen ändern
<code>fowner</code>	Berechtigungsprüfungen übergehen, inode-Flags und ACLs setzen
<code>net_bind_service</code>	Socket an privilegierten Port (>1024) binden
<code>net_admin</code>	Netzwerk und Firewall konfigurieren, Promiscuous Mode, Multitasking
<code>net_raw</code>	Raw packages versenden, transparente Proxys
<code>sys_module</code>	Kernel module laden und entfernen
<code>sys_chroot</code>	<code>chroot</code> -Umgebungen nutzen
<code>sys_ptrace</code>	Prozesse mit <code>ptrace</code> analysieren
<code>sys_admin</code>	Vollumfängliche Systemadministration, <b>nicht verwenden</b>
<code>sys_nice</code>	Prozessprioritäten erhöhen

# NETZWERK-REGELUNGEN

Flag	Beschreibung
<code>network,</code>	Generell erlaubt
<code>deny network,</code>	Generell verboten
<code>network tcp,</code>	Nur TCP erlaubt
<code>network udp,</code>	Nur UDP erlaubt
<code>network inet stream,</code>	Nur IPv4 erlaubt
<code>network inet6 dgram,</code>	Nur IPv6 erlaubt
<code>network (read, write, bind) inet stream</code>	Lesen/Schreiben/Lauschen unter IPv4 erlaubt

# LAB 10

## APPARMOR-PROFIL ERSTELLEN

# LAB 11

## APPARMOR VERWALTEN

# ZUSAMMENFASSUNG

- > steuert ebenfalls Prozesse auf **Systemebene**
  - > Ausnahmen werden explizit **pro Prozess** aktiviert
- > implementiert **Access Control** u.a. für
  - > Dateien, Capabilities, Netzwerk, Mounts, DBUS und Unix Sockets

# ZUSAMMENFASSUNG

- > steuert ebenfalls Prozesse auf **Systemebene**
  - > Ausnahmen werden explizit **pro Prozess** aktiviert
- > implementiert **Access Control** u.a. für
  - > Dateien, Capabilities, Netzwerk, Mounts, DBUS und Unix Sockets
- > **Profile** definieren Berechtigungen einer Anwendung
  - > Anwendung darf nur erlaubte Aufrufe tätigen
  - > können modular gehalten werden, um die Pflege zu erleichtern
  - > **Flags** geben Berechtigungen an

# ZUSAMMENFASSUNG

- > steuert ebenfalls Prozesse auf **Systemebene**
  - > Ausnahmen werden explizit **pro Prozess** aktiviert
- > implementiert **Access Control** u.a. für
  - > Dateien, Capabilities, Netzwerk, Mounts, DBUS und Unix Sockets
- > **Profile** definieren Berechtigungen einer Anwendung
  - > Anwendung darf nur erlaubte Aufrufe tätigen
  - > können modular gehalten werden, um die Pflege zu erleichtern
  - > **Flags** geben Berechtigungen an
- > Variablen werden als **Tunables** gespeichert
- > Mit **Abstractions** existieren wiederverwendbare Vorlagen

// OPENVAS

# OPENVAS

- > **Open Vulnerability Assessment System**
- > Komponente zum Scannen nach bekannten Verwundbarkeiten
- > Fork von **Nessus**
  - > Lizenzänderung in 2005 von Tenable Networks (*closed-source*)

# OPENVAS

- > **Open Vulnerability Assessment System**
- > Komponente zum Scannen nach bekannten Verwundbarkeiten
- > Fork von **Nessus**
  - > Lizenzänderung in 2005 von Tenable Networks (*closed-source*)
- > Teil von **Greenbone Vulnerability Manager (GVM)**
  - > Software-Framework für Verwundbarkeitsmanagement
  - > Entwicklung von Greenbone Networks GmbH
- > Wird häufig im Zusammenhang mit **Penetrationstests** verwendet

# FUNKTIONEN

- > Überprüft Komponenten auf bekannte **Sicherheitslücken**, z.B.:
  - > Web-Anwendungen
  - > Server-Dienste

# FUNKTIONEN

- > Überprüft Komponenten auf bekannte **Sicherheitslücken**, z.B.:
  - > Web-Anwendungen
  - > Server-Dienste
- > Analysiert **Netztraffic**, um verwendete Komponenten zu **erraten**, z.B.:
  - > Webserver-Version
  - > verwendetes Betriebssystem und damit verbundene Sicherheitslücken

# FUNKTIONEN

- > Überprüft Komponenten auf bekannte **Sicherheitslücken**, z.B.:
  - > Web-Anwendungen
  - > Server-Dienste
- > Analysiert **Netztraffic**, um verwendete Komponenten zu **erraten**, z.B.:
  - > Webserver-Version
  - > verwendetes Betriebssystem und damit verbundene Sicherheitslücken
- > **externe** Scans ohne Zugriff auf das Betriebssystem eines Hosts
- > Es können jedoch auch Zugangsdaten hinterlegt werden, um Scan-Ergebnisse zu konkretisieren

# FUNKTIONEN

- > Bekannte Sicherheitslücken werden täglich über einen **Feed** aktualisiert:
  - > **NVT** (*Network Vulnerability Tests*), ca. 50.000 derzeit verfügbar
  - > **CVE** (*Common Vulnerabilities and Exposures, internationale Datenbank für Sicherheitslücken*)
  - > **CPE** (*Common Platform Enumeration, Namenskonvention*)
  - > **OVAL** (*Open Vulnerability and Assessment Language, Beschreibungsstandard für Schwachstellenmanagement*)

# FUNKTIONEN

- > Steuerung über Web-Oberfläche
- > Zugriff auf diese benötigt einen **validen Benutzer**
- > Benutzer könnten lokal oder via LDAP gepflegt werden
- > Gruppen- und rollenfähig

# FUNKTIONEN

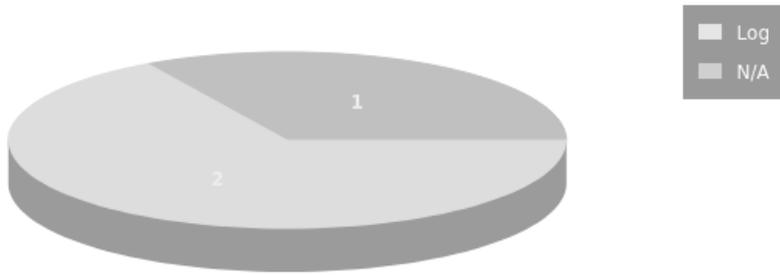
- > Steuerung über Web-Oberfläche
- > Zugriff auf diese benötigt einen **validen Benutzer**
- > Benutzer könnten lokal oder via LDAP gepflegt werden
- > Gruppen- und rollenfähig
- > Auf Basis definierter Richtlinien können **Audits** erstellt werden
  - > z.B. IT-Grundschutz Kompendium
- > Erstellen von **Tickets** zur Dokumentation der Härtung



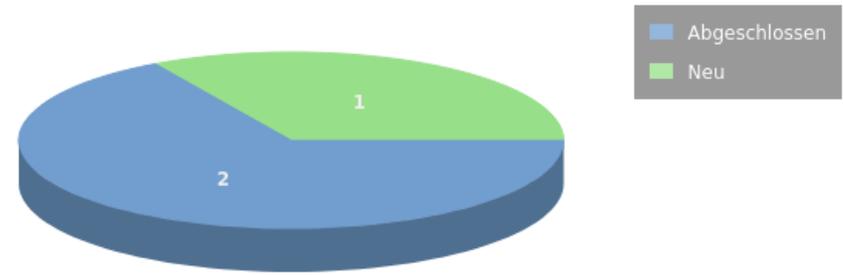
## Dashboards

Overview

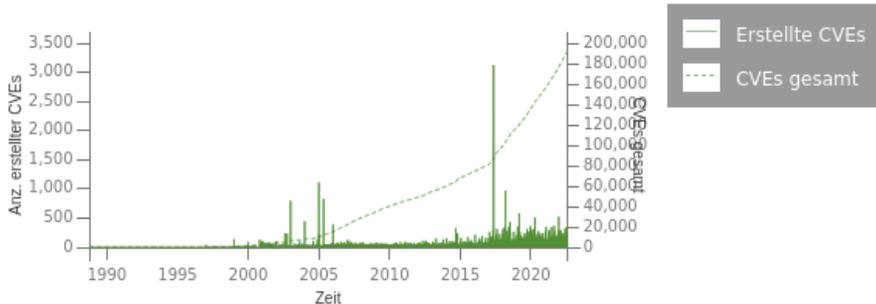
Aufgaben nach Schweregradklasse (Gesamt: 3)



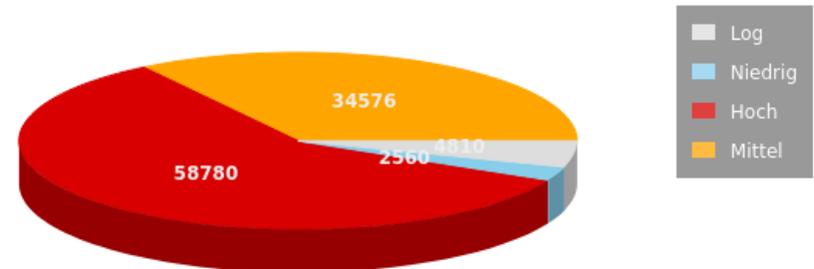
Aufgaben nach Status (Gesamt: 3)



CVEs nach Erstellungszeit



NVTs nach Schweregradklasse (Gesamt: 100726)



# SCANS AUSFÜHREN

- > Zu überprüfende Hosts werden als **Ziele** definiert
  - > manuelle Definition oder Datei-Import
- > Scans können einmalig oder **wiederkehrend** ausgeführt werden
  - > sinnvoll für regelmäßige interne Auditierung

# SCANS AUSFÜHREN

- > Zu überprüfende Hosts werden als **Ziele** definiert
  - > manuelle Definition oder Datei-Import
- > Scans können einmalig oder **wiederkehrend** ausgeführt werden
  - > sinnvoll für regelmäßige interne Auditierung
- > Start der Aufgabe
- > Während Ausführung wird ein **Bericht** erstellt
  - > enthält detaillierte Informationen über **Findings**

## Neue Aufgabe



**Name**

**Kommentar**

**Scan-Ziele**



**Benachrichtigungen**



**Zeitplan**



Einmalig

**Ergebnisse zu  
Assets  
hinzufügen**

Ja  Nein

**Übersteuerungen  
anwenden**

Ja  Nein

**Min. QdE**

%

**Änderbare  
Aufgabe**

Ja  Nein

**Berichte  
automatisch  
löschen**

Berichte nicht automatisch löschen  
 Älteste Berichte automatisch löschen, aber neuesten Bericht behalten  Berichte

**Scanner**



Abbrechen

Speichern



Filter




**Bericht: Mi., 28. Sep. 2022 08:51 UTC**

Abgeschlossen

ID: 0bb0ebdf-32b6-4c19-bcfb-f5b6190ee19a

Erstellt: Mi., 28. Sep. 2022 08:51 UTC

Geändert: Mi., 28. Sep. 2022 09:07 UTC

Besitzer: admin

- Informationen
- Ergebnisse  
(5 von 48)
- Hosts  
(1 von 1)
- Ports  
(2 von 2)
- Anwendungen  
(2 von 2)
- Betriebssysteme  
(1 von 1)
- CVEs  
(2 von 2)
- Geschlossene CVEs  
(0 von 0)
- TLS-Zertifikate  
(0 von 0)
- Fehlermeldungen  
(0 von 0)
- Benutzer-Tags  
(0)

◀◀ 1 - 5 von 5 ▶▶

Schwachstelle	🔧	Schweregrad ▼	QdE	Host		Ort	Erstellt
				IP	Name		
<a href="#">SSH Brute Force Logins With Default Credentials Reporting</a>	↔	7.5 (Hoch)	95 %	192.168.56.20	0-node1.sva.de	22/tcp	Mi., 28. Sep. 2022 09:07 UTC
<a href="#">HTTP Debugging Methods (TRACE/TRACK) Enabled</a>	↔	5.8 (Mittel)	99 %	192.168.56.20	0-node1.sva.de	80/tcp	Mi., 28. Sep. 2022 08:56 UTC
<a href="#">Weak Key Exchange (KEX) Algorithm(s) Supported (SSH)</a>	↔	5.3 (Mittel)	80 %	192.168.56.20	0-node1.sva.de	22/tcp	Mi., 28. Sep. 2022 08:55 UTC
<a href="#">Weak Encryption Algorithm(s) Supported (SSH)</a>	↔	4.3 (Mittel)	95 %	192.168.56.20	0-node1.sva.de	22/tcp	Mi., 28. Sep. 2022 08:55 UTC
<a href="#">TCP timestamps</a>	↔	2.6 (Niedrig)	80 %	192.168.56.20	0-node1.sva.de	general/tcp	Mi., 28. Sep. 2022 08:55 UTC

(Angewandter Filter: apply\_overrides=0 levels=hml rows=100 min\_qod=70 first=1 sort-reverse=severity)

◀◀ 1 - 5 von 5 ▶▶

# LAB 12

## OPENVAS HOST-AUDIT

# ZUSAMMENFASSUNG

- > **Scanner** wie OpenVAS können beim Aufdecken von **Verwundbarkeiten** helfen
  - > oft im Zusammenhang mit **Penetrationstests** verwendet

# ZUSAMMENFASSUNG

- > **Scanner** wie OpenVAS können beim Aufdecken von **Verwundbarkeiten** helfen
  - > oft im Zusammenhang mit **Penetrationstests** verwendet
- > Software-**Framework** zur Ausführung umfangreicher Überprüfungen
  - > überprüft u.a. Server-Dienste auf **Sicherheitslücken**
  - > kann **Netztraffic** analysieren, um verwendete Komponenten zu erraten
  - > startet i.d.R. ohne Interaktion mit den betroffenen Systemen

# ZUSAMMENFASSUNG

- > **Scanner** wie OpenVAS können beim Aufdecken von **Verwundbarkeiten** helfen
  - > oft im Zusammenhang mit **Penetrationstests** verwendet
- > Software-**Framework** zur Ausführung umfangreicher Überprüfungen
  - > überprüft u.a. Server-Dienste auf **Sicherheitslücken**
  - > kann **Netztraffic** analysieren, um verwendete Komponenten zu erraten
  - > startet i.d.R. ohne Interaktion mit den betroffenen Systemen
- > Zu überprüfende Hosts werden als **Ziele**
  - > gefundene Lücken können dokumentiert und verwaltet werden
  - > **Scans** können **wiederkehrend** stattfinden

// FAIL2BAN

# FAIL2BAN

- > Kombiniertes **IDS** / **IPS\*** zum Schutz vor Bruteforcing
- > in Python geschriebenes Framework, auf **POSIX**-Systemen lauffähig

# FAIL2BAN

- > Kombiniertes **IDS / IPS\*** zum Schutz vor Bruteforcing
- > in Python geschriebenes Framework, auf **POSIX**-Systemen lauffähig
- > Integriert sich in:
  - > Firewalls
  - > Mail-Server
  - > Webserver
  - > Datenbanken

\* Intrusion **D**etection/**P**revention **S**ystem

# FAIL2BAN

- > **Filter** überwachen **Log-Dateien** aktivierter Dienste anhand
  - > **Sucheinträgen** und Schlagwörtern
  - > regulärer Ausdrücke

# FAIL2BAN

- > **Filter** überwachen **Log-Dateien** aktivierter Dienste anhand
  - > **Sucheinträgen** und Schlagwörtern
  - > regulärer Ausdrücke
- > **Schwellwerte** steuern, ab wann IP-Adressen gesperrt werden
  - > z.B. nach 3 Versuchen
- > **Actions** (*hinterlegte Skripte*) können IP-Adressen auf mehrere Arten sperren
  - > z.B. via `nftables`

# FAIL2BAN

- > **Filter** überwachen **Log-Dateien** aktivierter Dienste anhand
  - > **Sucheinträgen** und Schlagwörtern
  - > regulärer Ausdrücke
- > **Schwellwerte** steuern, ab wann IP-Adressen gesperrt werden
  - > z.B. nach 3 Versuchen
- > **Actions** (*hinterlegte Skripte*) können IP-Adressen auf mehrere Arten sperren
  - > z.B. via `nftables`
- > Die Kombination eines *Filters* und einer *Action* wird **Jail** genannt
  - > z.B. nach 3 fehlerhaften SSH-Loginversuchen über `nftables` sperren

# ACTIONS

- > Es werden zahlreiche vorkonfigurierte Actions ausgeliefert - u.a.:
  - > bsdfw, firewall-cmd, ipfw, iptbales, nftables, osx-ipfw, shorewall, ufw, npf
  - > cloudflare, dshield, netscaler
  - > mail, sendmail
  - > abuseip, badip
- > Definitionen unterhalb `/etc/fail2ban/action.d`

# JAILS

- > Es gibt zahlreiche Konfigurationsbeispiele - u.a.:
  - > `sshd`, `dropbear`, `Guacamole`, `Webmin`, `phpMyAdmin`
  - > `Apache`, `NGINX`, `lighttpd`, `Roundcube`, `Horde`, `Drupal`
  - > `proftpd`, `vsftpd`
  - > `Courier`, `Dovecot`, `Postfix`, `Sendmail`, `Exim`, `Cyrus`
  - > `MySQL`, `MongoDB`
  - > `GitLab`

# JAILS

- > Es gibt zahlreiche Konfigurationsbeispiele - u.a.:
  - > `sshd`, `dropbear`, `Guacamole`, `Webmin`, `phpMyAdmin`
  - > `Apache`, `NGINX`, `lighttpd`, `Roundcube`, `Horde`, `Drupal`
  - > `proftpd`, `vsftpd`
  - > `Courier`, `Dovecot`, `Postfix`, `Sendmail`, `Exim`, `Cyrus`
  - > `MySQL`, `MongoDB`
  - > `GitLab`
- > Konfiguration unterhalb `/etc/fail2ban/jail.d`
- > Beispiele unter `/etc/fail2ban/jail.conf`

# MAILREPORTS UND MONITORING

- > fail2ban kann über Aktivitäten via Mail informieren
- > Für Monitoringsysteme gibt es Plugins:
  - > Nagios-kompatibel: [\[klick!\]](#)
  - > check\_mk: [\[klick!\]](#)
  - > Zabbix: [\[klick!\]](#)

# LAB 13

## FAIL2BAN EINSETZEN

# ZUSAMMENFASSUNG

- > Intrusion Detection/Prevention Systeme können vor **Bruteforcing** schützen
- > **fail2ban** integriert sich in verschiedene **Komponenten**
  - > Firewalls, Mail-Server, Webserver, Datenbanken und Applikationen

# ZUSAMMENFASSUNG

- > Intrusion Detection/Prevention Systeme können vor **Bruteforcing** schützen
- > **fail2ban** integriert sich in verschiedene **Komponenten**
  - > Firewalls, Mail-Server, Webserver, Datenbanken und Applikationen
- > **Filter** kontrollieren **Log-Dateien** überwachter Dienste
  - > Suche anhand **Schlagwörtern** und regulärer Ausdrücke
- > **Actions** dienen zur Sperrung nach erreichten **Schwellwerten**

# ZUSAMMENFASSUNG

- > Intrusion Detection/Prevention Systeme können vor **Bruteforcing** schützen
- > **fail2ban** integriert sich in verschiedene **Komponenten**
  - > Firewalls, Mail-Server, Webserver, Datenbanken und Applikationen
- > **Filter** kontrollieren **Log-Dateien** überwachter Dienste
  - > Suche anhand **Schlagwörtern** und regulärer Ausdrücke
- > **Actions** dienen zur Sperrung nach erreichten **Schwellwerten**
- > Das Zusammenspiel aus Filter und Actions wird auch **Jail** genannt
- > Reporting via Mail oder Integration in gängige Monitoring-Lösungen

// DEV-SEC

# MOTIVATION

- > Eine regelmäßige Auditierung der Systeme ist **unabdingbar**
  - > Ständige Updates bringen **neue** Komponenten und **Fehler**
  - > Kritische Lücken erfordern **außerplanmäßiges** Handeln (*Heartbleed, Spectre, Meltdown*)

# MOTIVATION

- > Eine regelmäßige Auditierung der Systeme ist **unabdingbar**
  - > Ständige Updates bringen **neue** Komponenten und **Fehler**
  - > Kritische Lücken erfordern **außerplanmäßiges** Handeln (*Heartbleed, Spectre, Meltdown*)
- > Sicherheitsstandards sind **essentiell** für (*Cloud-*)Anbieter
  - > Siehe beispielsweise [BSI C5-Anforderungskatalog](#)

# MOTIVATION

- > Eine regelmäßige Auditierung der Systeme ist **unabdingbar**
  - > Ständige Updates bringen **neue** Komponenten und **Fehler**
  - > Kritische Lücken erfordern **außerplanmäßiges** Handeln (*Heartbleed, Spectre, Meltdown*)
- > Sicherheitsstandards sind **essentiell** für (*Cloud-*)Anbieter
  - > Siehe beispielsweise [BSI C5-Anforderungskatalog](#)
- > System muss schon bei der Bereitstellung **möglichst sicher** sein
  - > **Nachträgliches** Absichern erfolgt erfahrungsgemäß nicht
- > Hardening muss weitestgehend automatisiert werden > **praktikabel** und effizient

# DEV-SEC

- > Nicht nur die sichere Bereitstellung neuer Systeme ist wichtig
- > Auch **bestehende** Systeme müssen **regelmäßig** überprüft und abgesichert werden
  - > Je nach Systemlandschaft hoher Aufwand

# DEV-SEC

- > Nicht nur die sichere Bereitstellung neuer Systeme ist wichtig
- > Auch **bestehende** Systeme müssen **regelmäßig** überprüft und abgesichert werden
  - > Je nach Systemlandschaft hoher Aufwand
- > Dev-Sec auditiert und sichert Server und Applikationen ab
- > **Baselines** u.a. für
  - > Microsoft Windows / Linux
  - > SSH / SSL
  - > Apache / NGINX

# DEV-SEC

- > Integriert sich in Puppet, Chef und Ansible
  - > Neue und bestehende Systeme können so **automatisiert** angepasst werden

# DEV-SEC

- > Integriert sich in Puppet, Chef und Ansible
  - > Neue und bestehende Systeme können so **automatisiert** angepasst werden
- > Grundlagen für Dev-Sec-Kataloge:
  - > **CIS**-Benchmarks (*Center for Internet Security*)
  - > NSA Hardening Guides
  - > Hersteller Best-Practices und Security Hardening Guides

# DEV-SEC

- > Integriert sich in Puppet, Chef und Ansible
  - > Neue und bestehende Systeme können so **automatisiert** angepasst werden
- > Grundlagen für Dev-Sec-Kataloge:
  - > **CIS**-Benchmarks (*Center for Internet Security*)
  - > NSA Hardening Guides
  - > Hersteller Best-Practices und Security Hardening Guides
- > Überprüfung der Systeme via **InSpec**
- > Open-Source: [[klick!](#)]

# INSPEC

- > Framework für Audits und Tests
  - > Compliance
  - > Security
  - > Policy-Anforderungen
  - > Unit-Tests
  - > ...

# INSPEC

- > Framework für Audits und Tests
  - > Compliance
  - > Security
  - > Policy-Anforderungen
  - > Unit-Tests
  - > ...
- > Selbstsprechende Definition von Anforderungen, einfach zu verstehen
- > Teil von Chef, aber OSS: [\[klick!\]](#)

# BEISPIEL

```
describe package('telnetd') do
  it { should_not be_installed }
end
```

- > Definition einer `package`-Ressource
- > Referenzieren eines bestimmten Namen
- > Bedingung, dass Paket nicht installiert sein soll

# BASELINES

- > Baselines entsprechen **InSpec**-Profilen für:
  - > Microsoft Windows / Linux
  - > SSH / SSL
  - > Docker / Kubernetes
  - > Apache / NGINX
  - > MySQL / PostgreSQL

# BASELINES

- > Baselines entsprechen **InSpec**-Profilen für:
  - > Microsoft Windows / Linux
  - > SSH / SSL
  - > Docker / Kubernetes
  - > Apache / NGINX
  - > MySQL / PostgreSQL
- > Noch nicht offizielle Baselines:
  - > [Windows Patch](#) / [Linux Patch](#)
  - > [OpenStack](#)

# REMEDICATION

- > Für die meisten\* Baselines gibt es vordefinierte Automatismen für:
  - > Ansible (Rolle)
  - > Chef (Cookbook)
  - > Puppet (Modul)

# REMEDIATION

- > Für die meisten\* Baselines gibt es vordefinierte Automatismen für:
  - > Ansible (Rolle)
  - > Chef (Cookbook)
  - > Puppet (Modul)
- > Download über:
  - > [Dev-Sec-Webseite](#)
  - > [GitHub](#)
  - > jeweilige Tool-Community (*Ansible Galaxy, Puppetforge,...*)

\* ausgenommen Docker und Kubernetes

# REMEDIATION

## Beispielhaftes Playbook:

```
- name: Harden servers
  hosts: node1
  become: true
  roles:
    - name: devsec.hardening.os_hardening
      sysctl_overwrite:
        # Enable IPv4 forwarding (needed for containers)
        net.ipv4.ip_forward: 1
```

## Herunterladen der Collection:

```
$ ansible-galaxy collection install devsec.hardening
```

# REMEDIATION

## Ausführen des Playbooks:

```
$ ansible-playbook hardening.yml
PLAY [Harden servers] *****
...
TASK [devsec.hardening.os_hardening : Configure selinux | selinux-01] ****
ok: [node1]

PLAY RECAP *****
node1          : ok=53    changed=1    unreachable=0    failed=0    sl
```

# BEST PRACTICES

- > Rollen zuerst auf Test-Systemen ausführen und **ausgiebig testen**
  - > Anwendungen könnten nach Härtung nicht mehr funktionieren
- > Dokumentation und insbesondere Parameter/Schalter konsultieren
  - > diese stellen i.d.R. einzelne Härtungen ab

# BEST PRACTICES

- > Rollen zuerst auf Test-Systemen ausführen und **ausgiebig testen**
  - > Anwendungen könnten nach Härtung nicht mehr funktionieren
- > Dokumentation und insbesondere Parameter/Schalter konsultieren
  - > diese stellen i.d.R. einzelne Härtungen ab
- > **Version-Pinning** - nie ungetestet neue Rollen verwenden
  - > `requirements.yml` nutzen!
  - > neue Härtungen könnten wieder Fehler nach sich ziehen
  - > bei neueren Versionen wieder zuerst auf Test-Systemen testen

# BEST PRACTICES

Beispielhafte `requirements.yml`:

```
---  
collections:  
  - name: devsec.hardening  
    version: 7.8.0
```

Installation benötigter Ansible-Inhalte:

```
$ ansible-galaxy collection install -r requirements.yml
```

Sofern auch Rollen definiert wurden:

```
$ ansible-galaxy role install -r requirements.yml
```

# LAB 14

## ANWENDEN VON DEV-SEC

# LAB 15

## ERSTELLEN EINES INSPEC-PROFILS

# ANSIBLE-LOCKDOWN

- > ähnlich zu Dev-Sec, jedoch ausschließlich für Ansible
- > benutzt [Goss](#) statt InSpec für **Auditierung**

# ANSIBLE-LOCKDOWN

- > ähnlich zu Dev-Sec, jedoch ausschließlich für Ansible
- > benutzt [Goss](#) statt InSpec für **Auditierung**
- > **CIS**- und **DISA**-Inhalte für verschiedene Betriebssysteme und Anwendungen
  - > Red Hat Enterprise Linux 7 bis 9
  - > Debian 11 sowie Ubuntu 18.04 bis 22.04
  - > Windows 10/11 sowie 2016 bis 2022
  - > Apache
  - > Postgres 12
  - > Kubernetes 1.6
- > auf [GitHub](#) verfügbar

# ZUSAMMENFASSUNG

- > Systeme müssen **regelmäßig auditiert** werden
  - > ständige Updates bringen **neue** Komponenten und somit auch **Lücken**
- > Hardening muss weitestgehend automatisiert werden > **praktikabel**

# ZUSAMMENFASSUNG

- > Systeme müssen **regelmäßig auditiert** werden
  - > ständige Updates bringen **neue** Komponenten und somit auch **Lücken**
- > Hardening muss weitestgehend automatisiert werden > **praktikabel**
- > Dev-Sec **automatisiert** das Auditieren und Absichern von Systemen
- > vorgefertigte **Baselines** und Härtungs-Automatismen u.a. für
  - > Linux, SSH, Apache
  - > Inhalte auf Basis von CIS-Benchmarks und NSA-Hardening Guides
- > Ausgiebiges **Testen** unabdingbar
- > `ansible-lockdown` ist eine weitere Alternative

# // LINKLISTE

- > [Liste gängiger Exploit-Datenbanken](#)
- > [SELinux-Wiki](#)
- > [SELinux Coloring Book](#)
- > [Anleitung zum Erstellen eigener SELinux-Policies](#)
- > [AppArmor Core Policy Reference](#)
- > [AppArmor Tutorial für Ubuntu](#)
- > [AppArmor unter Ubuntu debuggen](#)

DANKE FÜR EURE AUFMERKSAMKEIT!  
(JETZT BITTE WIEDER AUFWACHEN)

