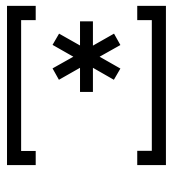

SELINUX

CARSTEN STROTMANN

Version 1.1, 27.10.2022

INHALTSVERZEICHNIS

CHAPTER 1. LINUX SECURITY MODULES UND SELINUX

- SELinux und Linux Security Module

CHAPTER 2. AUDIT SUBSYSTEM

- Das Linux Audit Subsystem

2.1. INSTALLATION

- Wir arbeiten auf den virtuellen Maschinen im Internet
- Audit-Subsystem Programme installieren. Das Paket `audit` ist in der Regel schon vorinstalliert. `audispd-plugins` enthält die Plugins des Audit-Dispatchers

```
dnf install audit audispd-plugins
```

2.2. KONFIGURATION

- Passe die Audit-Daemon Konfiguration an. Benutze hierfür die Informationen aus den Folien

```
$EDITOR /etc/audit/auditd.conf
```

- Audit-Dämon anschalten (so dass er bei einem Restart neu gestartet wird) und prüfen das der Prozess läuft

```
systemctl enable --now auditd
systemctl status auditd
journalctl -u auditd
```

- Dem Audit-Daemon ein "Hangup (HUP)" Signal senden um die Konfiguration neu zu laden

```
pkill -HUP auditd
```

- Audit-Daemon starten und stoppen

```
service auditd stop
service auditd start
```

- Red Hat Bug-Report: Unable to restart/stop auditd service using systemctl command in RHEL7: https://access.redhat.com/solutions/2664811

2.3. AD-HOC TRACE VON PROGRAMMEN

- Starte einen Audit-Subsystem Trace eines Prozesses (hier 4 x ICMP Echo per `ping`)

```
autrace /bin/ping -c 4 8.8.8.8
```

- Audit Log für einen speziellen `autrace` Aufruf anschauen. Die genaue Kommandozeile wurde vom `autrace` Programm ausgegeben.

```
ausearch -i -p <audit-id>
```

- Die Log-Ausgaben können über eine Shell-Pipeline ausgewertet werden. In diesem Beispiel werden alle System-Calls des `autrace` Aufruf aufgelistet und nach Häufigkeit sortiert ausgeben:

```
ausearch -i -p <audit-id> | grep type=SYSCALL | cut -f 6 -d ' ' | sort | uniq -c | sort -n
```

- Wurde der Hostname als Metadaten bei den Audit-Meldungen mit ausgegeben Konfiguration des Audit-Daemon), so muss per `cut` Befehl das Feld #7 ausgewertet werden (…`| cut -f 7 -d ' ' |`…)

2.4. RICHTLINIEN-DATEIEN

- Die vorinstallierte (leere) Richtlinien-Datei löschen

```
rm /etc/audit/rules.d/audit.rules
```

- Audit-Richtlinien Beispiele befinden sich unter `/usr/share/audit/sample-rules`

- Richtliniendateien aus den Beispielen in das Verzeichnis für die Audit-Regeln kopieren

```
cp /usr/share/audit/sample-rules/10-base-config.rules /etc/audit/rules.d/
cp /usr/share/audit/sample-rules/11-loginuid.rules /etc/audit/rules.d/
cp /usr/share/audit/sample-rules/30-stig.rules /etc/audit/rules.d/
cp /usr/share/audit/sample-rules/32-power-abuse.rules /etc/audit/rules.d/
cp /usr/share/audit/sample-rules/42-injection.rules /etc/audit/rules.d/
cp /usr/share/audit/sample-rules/43-module-load.rules /etc/audit/rules.d/
cp /usr/share/audit/sample-rules/70-einval.rules /etc/audit/rules.d/
cp /usr/share/audit/sample-rules/71-networking.rules /etc/audit/rules.d/
cp /usr/share/audit/sample-rules/99-finalize.rules /etc/audit/rules.d/
```

- Richtlinien für priviligierte Programme erstellen

```
cd /etc/audit/rules.d/
find /bin -type f -perm -04000 2>/dev/null | awk '{ printf "-a always,exit -F path=%s -F perm=x -F
auid>=1000 -F auid!=unset -F key=privileged\n", $1 }' > 31-privileged.rules
find /sbin -type f -perm -04000 2>/dev/null | awk '{ printf "-a always,exit -F path=%s -F perm=x -F
auid>=1000 -F auid!=unset -F key=privileged\n", $1 }' >> 31-privileged.rules
find /usr/bin -type f -perm -04000 2>/dev/null | awk '{ printf "-a always,exit -F path=%s -F perm=x
-F auid>=1000 -F auid!=unset -F key=privileged\n", $1 }' >> 31-privileged.rules
find /usr/sbin -type f -perm -04000 2>/dev/null | awk '{ printf "-a always,exit -F path=%s -F perm=x
-F auid>=1000 -F auid!=unset -F key=privileged\n", $1 }' >> 31-privileged.rules
filecap /bin 2>/dev/null | sed '1d' | awk '{ printf "-a always,exit -F path=%s -F perm=x -F
auid>=1000 -F auid!=unset -F key=privileged\n", $2 }' >> 31-privileged.rules
filecap /sbin 2>/dev/null | sed '1d' | awk '{ printf "-a always,exit -F path=%s -F perm=x -F
auid>=1000 -F auid!=unset -F key=privileged\n", $2 }' >> 31-privileged.rules
filecap /usr/bin 2>/dev/null | sed '1d' | awk '{ printf "-a always,exit -F path=%s -F perm=x -F
auid>=1000 -F auid!=unset -F key=privileged\n", $2 }' >> 31-privileged.rules
filecap /usr/sbin 2>/dev/null | sed '1d' | awk '{ printf "-a always,exit -F path=%s -F perm=x -F
auid>=1000 -F auid!=unset -F key=privileged\n", $2 }' >> 31-privileged.rules
```

- Die Richtlinie kompilieren und prüfen

```
augenrules --check
```

- Die kombinierte Audit-Richtlinien-Datei anschauen

```
less /etc/audit/audit.rules
```

- Die neue Policydatei in den Kernel laden

```
augenrules --load
```

- Aktive Audit-Regeln auflisten

```
auditctl -l
```

- Status des Audit-Subsystems anzeigen

```
# auditctl -s
enabled 2
failure 1
pid 12901
rate_limit 0
backlog_limit 8192
lost 0
backlog 0
backlog_wait_time 0
loginuid_immutable 0 unlocked
```

2.5. BEISPIELE FÜR AUDIT-ABFRAGEN

- Alle Audit-Einträge zum Thema "sudo" zeigen

```
ausearch -i -x sudo
```

- Report über fehlgeschlagende Anmeldeversuche

```
ausearch -m USER_AUTH,USER_ACCT --success no
```

- Alle Audit-Meldungen für Benutzer UID 1000

```
ausearch -ua 1000 -i
```

- Fehlgeschlagende Syscalls seit gestern

```
ausearch --start yesterday --end now -m SYSCALL -sv no -i
```

2.6. AUFGABE:

- Das Audit-Subsystem um neue Regel(n) zu Systemd erweitern:

  Alle Systemd-Konfigurationsdateien mit der Endung `*.conf`, welche direkt unter `/etc/systemd` (nicht in den Unterverzeichnissen) gespeichert sind, sollen auf Schreibzugriffe überwacht werden

  Die Audit-Events sollen mit dem Key `systemd` markiert werden

  Die neue Richtlinie kompilieren und aktivieren (ggf. Reboot notwendig)

  Die neue Richtlinie testen, in dem manuell an einer der überwachten Dateien eine Änderung vorgenommen wird

  Die Audit-Meldungen per `ausearch` anzeigen und analysieren (suchen nach Events mit dem Key `systemd`)

2.6.1. Beispiel-Lösung

- Erweiterung der Audit-Regeln

```
[...]
# Systemd Konfiguration
-w /etc/systemd/journald.conf -p wa -k systemd
-w /etc/systemd/logind.conf -p wa -k systemd
-w /etc/systemd/networkd.conf -p wa -k systemd
-w /etc/systemd/resolved.conf -p wa -k systemd
-w /etc/systemd/sleep.conf -p wa -k systemd
-w /etc/systemd/system.conf -p wa -k systemd
```

```
-w /etc/systemd/timesyncd.conf -p wa -k systemd
-w /etc/systemd/user.conf -p wa -k systemd
[...]
```

- Suche nach Audit-Events mit dem Schlüssel `systemd`

```
ausearch -i -k systemd
```

CHAPTER 3. SELINUX

- SELinux Grundlagen

3.1. SELINUX ERKUNDEN

- SELinux Hilfspakete installieren

```
dnf install policycoreutils setools libselinux-utils selinux-policy-doc setools-console \
    policycoreutils-python3 selinux-policy-devel policycoreutils-newrole
```

3.1.1. SELinux Label auf Dateien

- SELinux Label (Context) auf Dateien anzeigen

```
ls -lZ <pfad>
```

- Welche Dateien im Verzeichnis `/etc` sind vom SELinux Typ `system_conf_t`?
- Welchen SELinux Type haben neue Dateien im Heimverzeichnis des Benutzers `user`?
- Erstelle eine sortierte Liste der SELinux Datei-Typen (3tes Feld im SELinux Label user:role:type) im Verzeichnis `/usr/sbin`.

Lösungen

```
ls -lZ /usr/sbin/ | cut -f 3 -d ':' | cut -f 1 -d ' ' | sort | uniq -c | sort -n
```

3.1.2. SELinux Label auf Prozessen

- SELinux Label auf Prozessen anzeigen

```
ps -auxZ
```

3.1.3. SELinux Label auf dem aktuellen Benutzer anzeigen

```
# id -Z
```

3.1.4. SELinux Status abfragen

- Allgemeinen SELinux Status abfragen

```
sestatus
```

- Detaillierten SELinux Status abfragen

```
sestatus -v
```

- SELinux "enforcement" Status anzeigen

```
getenforce
```

- Verfügbare SELinux Module auflisten

```
semodule -l | less
```

- Die kompilierte (binäre) Richtlinien (Policy) Datei

```
ls -lh /etc/selinux/targeted/policy/
```

3.2. SELINUX FUNKTIONS-BEISPIEL

- Apache Webserver installieren und starten

```
dnf install httpd
systemctl enable --now httpd
```

- Kleine Webseite mit einem Editor anlegen ($EDITOR durch `vi`, `vim`, `nano`, `emacs` etc ersetzen)

```
$EDITOR /var/www/html/index.html
```

- Inhalt der HTML-Datei `/var/www/html/index.html`

```
<html>
<body>
<h1>Apache Webserver</h1>
</body>
</html>
```

- SELinux Security Context auf der Datei anzeigen

```
ls -lZ /var/www/html/index.html
```

- Port 80 in der Firewall erlauben

```
# firewall-cmd --zone=public --add-service=http --permanent
# firewall-cmd --reload
```

- Die Webseite sollte nun unter Port 80 (http, nicht https) mittels eines Webbrowsers abrufbar sein

3.2.1. Ein SELinux-Problem für den Apache Webserver erzeugen

- Eine Datei `index-html` Datei im Heim-Verzeichnis des Benutzers `root` erstellen und in das Apache-WWW-Verzeichnis **verschieben** (nicht kopieren):

```
rm /var/www/html/index.html
$EDITOR /root/index.html
mv /root/index.html /var/www/html/index.html
ls -lZ /var/www/html/index.html
```

- Apache sollte nun nicht mehr in der Lage sein, die HTML-Datei auszuliefern (Default-Apache 2 Webseite erscheint)
- SELinux LSM-Meldungen im Audit-Log zum Prozess `httpd` anzeigen (Modul `avc` = Access Vector Cache)

```
ausearch -m avc -ts recent -c httpd -i
```

- SELinux Sicherheits-Kontext der Datei prüfen

```
matchpathcon -V /var/www/html/index.html
```

- Es wird angezeigt das der SELinux Sicherheits-Kontext der Datei nicht korrekt ist. SELinux blockiert daher die Zugriffe vom Apache Webserver auf diese Datei

3.2.2. Das SELinux Problem mit dem Apache Webserver lösen

- SELinux Security Context für Apache-Dateien anzeigen

```
sesearch --allow --source httpd_t --target httpd_sys_content_t --class file
```

- SELinux Sicherheits-Kontext anpassen (manuell mit dem Befehl `chcon` (Change Context)

```
chcon --type httpd_sys_content_t /var/www/html/index.html
```

- (Alternativ) SELinux Sicherheits-Kontext aus der SELinux Policy angleichen

```
restorecon -v /var/www/html/index.html
```

3.3. AUFGABE: APACHE AUF EINEM ANDEREN PORT ALS 80 ODER 443

- Ändere die Apache Konfiguration unter `/etc/httpd/conf/httpd.conf` so das der Apache auf Port 1235/TCP auf HTTP-Anfragen horcht (Konfigurationsparameter `Listen: 1235`)
- Versuche den Apache Webserver mittels systemd neu zu starten (`systemctl restart httpd`). Dies wird fehlschlagen, da `httpd` sich nicht auf Port 1235 binden kann.
- Schaue in per `ausearch` in die Audit-Logdatei nach SELinux Fehlern des Prozesses `httpd`
- Erlaubte Ports für den Apache HTTP auflisten

```
semanage port -l | grep http_port_t
```

- Benutze den Befehl `semanage port` um den Port `1235` für den Prozess `httpd` im SELinux freizuschalten und starte den Apache Webserver neu.
- Port `1235` in der Firewall freischalten
- Teste, on der Firefox-Browser unter der URL http://selinuxNNN.linux-sicherheit.org:1235/ die Webseite des Webservers sehen kann.

Beispiellösung

- Apache Konfiguration bearbeiten

```
$EDITOR /etc/httpd/conf/httpd.conf
```

- Apache Prozess neu starten. Dies sollte fehlschlagen, da SELinux die Benutzung von Port 1235 für den Prozess `httpd` verbietet
- SELinux Fehler für `httpd` im Audit-Log suchen

```
ausearch -m avc -c httpd -ts recent -i
```

- Port 1235 für Apache in der SELinux Richtlinie hinzufügen

```
semanage port -a -t http_port_t -p tcp 1235
```

- Neue Portdefinition prüfen

```
semanage port -l | grep http_port_t
```

- Apache Webserver neu starten

```
systemctl restart httpd
```

- Port 1235 in der Firewall freischalten

```
firewall-cmd --zone=public  --add-port=1235/tcp --permanent
firewall-cmd --reload
```

- Änderungen mit `semanage` sind persistent, die Port-Änderung ist unter `/var/lib/selinux/targeted/active/ports.local` abgespeichert:

```
# This file is auto-generated by libsemanage
# Do not edit directly.

portcon tcp 1235 system_u:object_r:http_port_t:s0
```

3.4. SELINUX RICHTLINIEN KONFIGURIEREN AM BEISPIEL BIND 9

- In dieser Aufgabe arbeiten wir mit dem BIND 9 DNS Server. Installation des BIND 9 Servers

```
# dnf install bind
```

- Starte den BIND 9 DNS Server

```
# systemctl enable --now named
```

- Der BIND 9 DNS kann interne Statistiken über das HTTP Protokoll ausliefern. Der Standardport für diese Funktion ist 8053.
- In der Datei `/etc/named.conf` füge die folgende Definition des Statistik-Channels hinzu (am Beginn der Konfigurationsdatei)

```
statistics-channels {
      inet * port 8053 allow { any; };
};
```

- Für Produktions-Installationen sollte statt `any` der Zugriff auf diesen Dienst per IP-Adressen ACL beschränkt werden
- Die Konfiguration prüfen und den BIND 9 DNS Server neu starten

```
# named-checkconf
# systemctl restart named
```

- Im Journal-Log des BIND 9 DNS Servers sollte eine Meldung auftauchen, das der Port 8053 nicht geöffnet werden konnte

```
# journalctl -u named
```

```
   [...]
   Oct 17 18:08:45 selinux22 named[34204]: /etc/named.conf:11: couldn't allocate statistics channel
0.0.0.0#8053: permission denied
```

- Der Fehler im Audit-Log

```
 # ausearch -m avc -ts recent  -i
----
type=PROCTITLE msg=audit(09/17/2021 04:06:05.836:2278) : proctitle=/usr/sbin/named -u named -c
/etc/nam
type=SYSCALL msg=audit(09/17/2021 04:06:05.836:2278) : arch=x86_64 syscall=bind success=no
    exit=EACCES(Permission denied) a0=0x15 a1=0x7f5183d24660 a2=0x10 a3=0x7f5183d244fc
    items=0 ppid=111613 pid=111615 auid=unset uid=named gid=named euid=named suid=named
    fsuid=named egid=named sgid=named fsgid=named tty=(none) ses=unset
    comm=isc-worker0000 exe=/usr/sbin/named subj=system_u:system_r:named_t:s0 key=(null)
type=AVC msg=audit(09/17/2021 04:06:05.836:2278) : avc:  denied  { name_bind } for  pid=111615
    comm=isc-worker0000 src=8053 scontext=system_u:system_r:named_t:s0
    tcontext=system_u:object_r:unreserved_port_t:s0
    tclass=tcp_socket permissive=0
```

3.4.1. Setzen des Boolean, um BIND 9 den Zugriff auf die http-Ports zu erlauben

- Das Umschalten des SELinux-Boleans `named_tcp_bind_http_port` erlaubt den Zugriff auf Ports, die für den Typ `http_port_t` definiert sind

```
   # setsebool named_tcp_bind_http_port=on
```

- Aber Port 8053 ist nicht unter diesen Ports

```
   # semanage port -l | grep http_port_t
   http_port_t tcp     80, 81, 443, 488, 8008, 8009, 8443, 9000
```

- Lösung 1: Benutze einen Port für den Statistik-Challen welcher in der SELinux Policy für `http_port_t` schon erlaubt ist. In der `named.conf`:

```
    statistics-channels {
      inet * port 8008 allow { any; };
    };
```

- Lösung 2: Füge den BIND 9 Statistik-Port der Liste der Port für `http_port_t` hinzu:

```
   # semanage port -a -t http_port_t -p tcp 8053
   # semanage port -l | grep http_port_t
   http_port_t                    tcp  8053, 80, 81, 443, 488, 8008, 8009, 8443, 9000
```

3.5. AUFGABE: EINE DYNAMISCHE DNS ZONE

- Erstelle auf der VM eine neue DNS Zonendatei für den DNS Server unter `/var/named/example.org`. Der Inhalt dieser Datei (`NNN` durch die eigene Teilnehmer-Nummer ersetzen):

```
$ttl 3600
@      IN SOA selinuxNNN.linux-sicherheit.org.  .  1001  2h 1h 40d 1h
       IN NS  selinuxNNN.linux-sicherheit.org.
       IN A   1.2.3.4
```

- Prüfe den SELinux Sicherheit Kontext auf der neuen Zonendatei:

```
% ls -Z /var/named/example.org
unconfined_u:object_r:named_zone_t:s0 /var/named/example.org
```

- Erstelle eine neue Zonen-Definition in der BIND 9 Konfigurations-Datei `/etc/named.conf`

```
zone "example.org" {
     type master;
     file "example.org";
};
```

- Prüfe die neue Konfiguration und lade die Konfiguration in den BIND 9 DNS Server

```
% named-checkconf -z
% rndc reconfig
% rndc zonestatus example.org
```

- Editiere die BIND 9 Konfiguraitonsdatei `/etc/named.conf` und verwandle die Zone `example.com` in eine dynamsche Zone (eine Zone welche nicht mehr per Editor verändert wird, sondern über das DNS Protokoll aus dem Netzwerk):

```
zone "example.org" {
     type master;
     file "example.org";
     update-policy local;
     };
```

- Prüfe die Konfiguration mit `named-checkconf -z`, lade die Konfiguration und prüfe den Status der Zone mittels `rndc zonestatus`. Die Zone sollte nun als eine dynamsiche Zone angezeigt werden.
- Erzeuge einen neuen IPv6-Adressen-Eintrag in der Zone (mittels `nsupdate` Programm):

```
% nsupdate -l
> ttl 3600
> add example.org in aaaa 2001:db8::1
> send
> quit
```

- Das dynamische Update schlägt fehl. Warum?
- Prüfe das BIND 9 Log:

```
% journalctl -eu named
```

- Prüfe das Audit-Log:

```
% ausearch -m avc -x /usr/sbin/named -i
```

- Wie können wir dieses Problem lösen?

3.5.1. Lösung 1

- Verschiebe die Zonen-Datein in das Verzeichnis `/var/named/dynamic` und passe den SELinux Datei-Kontext (und die Unix-Berechntigungen) an:

```
% mv example.org dynamic/
% chown named: dynamic/example.org
% restorecon -vr dynamic/
Relabeled /var/named/dynamic/example.org from unconfined_u:object_r:named_zone_t:s0 to
unconfined_u:object_r:named_cache_t:s0
```

- Passe die BIND 9 Konfiguration an:

```
zone "example.org" {
     type master;
     file "dynamic/example.org";
     update-policy local;
     };
```

3.5.2. Lösung 2

- Setze den SELinux Boolean Schalter `named_write_master_zones` auf `on`

```
% setsebool named_write_master_zones=on
```

3.6. AUFGABE: BIND 9 ZONENDATEI IN UNGEWÖHNLICHEM VERZEICHNIS

- In dieser Aufgabe wollen wir die BIND 9 Zonendateien aus operativen Gründen unterhalb des `/srv/` Filesystem-Pfad speichern
- Erstellen Sie das Verzeichnis `/srv/bind/zones/primary/`

```
% mkdir -p /srv/bind/zones/primary
```

- Erstellen Sie eine Zonendatei für die Zone `example.net` in diesem neuen Verzeichnis

```
$ttl 3600
@      IN SOA selinuxNNN.linux-sicherheit.org.  .  1001  2h 1h 40d 1h
       IN NS  selinuxNNN.linux-sicherheit.org.
       IN A   1.2.3.4
```

- Erstellen Sie einen neuen Zonen-Block ür die neue primäre Zone in der BIND 9 Konfigurationsdatei `/etc/named.conf`

```
zone "example.net" {
  type master;
  file "/srv/bind/zones/primary/example.net";
};
```

- Prüfe die Konfiguration, lade die Konfiguration in den BIND 9 DNS Server und prüfe den Zonen-Status der Zone `example.net`
- Die Zone wurde nicht geladen
- Suchen Sie mittels `journalctl` und `ausearch` nach den Gründen

```
% journalctl -eu named
% ausearch -m avc -x /usr/sbin/named -i
```

- Wie kann dieses SELinux Problem gelöst werden?

3.6.1. Lösung

- Ändern Sie die SELinux Policy für den BIND 9 Server und übernehmen sie das neue Verzeichnis als ein Verzeichnis für BIND 9 Zonendateien

```
% semanage fcontext -a -t named_zone_t --ftype f "/srv/bind/zones(/.*)?"
% semanage fcontext -a -t named_zone_t --ftype d "/srv/bind/zones(/.*)?"
```

- Wenden Sie die neuen SELinux Datei-Label auf die existierenden Zonendateien an

```
% restorecon -rv /srv/bind/zones/
Relabeled /srv/bind/zones from unconfined_u:object_r:var_t:s0 to
unconfined_u:object_r:named_zone_t:s0
Relabeled /srv/bind/zones/primary from unconfined_u:object_r:var_t:s0 to
unconfined_u:object_r:named_zone_t:s0
Relabeled /srv/bind/zones/primary/example.net from unconfined_u:object_r:var_t:s0 to
unconfined_u:object_r:named_zone_t:s0
```

3.7. AUFGABE: RNDC AUF PORT 65053

- In dieser Aufgabe wollen wir aus operativen Grünen den BIND 9 "remote name daemon control" Kanal auf von Port 953 (Default) auf Port 65053 ändern
- Konfigurieren Sie BIND 9 so das `rndc` API-Anfragen auf Port 65053 entgegen genommen werden. Fügen Sie die folgende Konfiguration der BIND 9 Konfigurationsdatei `/etc/named.conf` hinzu:

```
include "/etc/rndc.key";

controls {
       inet 127.0.0.1 port 65053
               allow { 127.0.0.1; } keys { "rndc-key"; };
 };
```

- Prüfen Sie die Konfiguration und laden Sie die neue Konfiguration in den BIND 9 DNS Server

```
% systemctl restart named
```

- Testen Sie die RNDC Funktion auf Port 65053

```
% rndc -p 65053 status
```

- Was ist hier das Problem? Benutzen Sie `journalctl` und `ausearch` zur Fehlersuche
- Wie können wir dieses Problem lösen?
- Hinweis:

```
% semanage port  -l | grep 953
```

3.7.1. Lösung:

- Fügen Sie Port 65053 zur SELinux Richtlinien-Definition des Types `rndc_port_t` hinzu

```
% semanage port -a -t rndc_port_t -p tcp 65053
```

- Starten Sie BIND 9 neu

```
% systemctl restart named
```

- Testen Sie die `rndc` Funktion

```
% rndc -p 65053 status
version: BIND 9.11.26-RedHat-9.11.26-4.el8_4 (Extended Support Version) <id:3ff8620>
running on selinux-test: Linux x86_64 4.18.0-305.3.1.el8_4.x86_64 #1 SMP Thu Jun 17 07:52:48 UTC
2021
boot time: Sun, 19 Sep 2021 19:26:35 GMT
last configured: Sun, 19 Sep 2021 19:26:35 GMT
```

```
configuration file: /etc/named.conf
CPUs found: 1
worker threads: 1
UDP listeners per interface: 1
number of zones: 105 (97 automatic)
debug level: 0
xfers running: 0
xfers deferred: 0
soa queries in progress: 0
query logging is OFF
recursive clients: 0/900/1000
tcp clients: 2/150
TCP high-water: 2
server is up and running
```

3.8. SELINUX - POLICY DEVELOPMENT

- Für dieses Kapitel arbeiten wir auf dem VM Maschinen
- Andere Web-Server (Apache/NGINX etc) auf der VM stoppen
- Für die Dauer dieser Übung die Firewall auf der VM deaktivieren

```
systemctl stop firewalld
```

3.8.1. SELinux Policy erstellen

- In diesem Kapitel werden wir eine SELinux Richtlinie (Policy) für einen Dienst entwickeln, welcher bisher noch nicht durch SELinux geschützt ist

  Dies kann in der Praxis für Software notwendig werden, welche von externen Entwicklern geliefert wurde oder im eigenen Hause entwickelt wird

  Unsere Beispiel-Anwendung ist ein sehr einfacher Webserver

- Um die Beispiel-Anwendung aus dem Quellcode übersetzen zu können installieren wir den GCC C-Compiler

```
dnf install gcc
```

- Für die Entwicklung neuer SELinux Richtlinien benötigen wir die Pakete für die SELinux Policy-Entwicklung (diese sind im Kurs ggf. schon installiert)

```
dnf install policycoreutils-python3 selinux-policy-devel
```

- Nach der Installation dieser Pakete befinden sie die SELinux Policy Quelldateien (der von Red Hat und der Community erstellen Richtlinien) im Verzeichnis `/usr/share/selinux/devel/`

```
  ls -l /usr/share/selinux/devel/
```

3.8.2. Ein neuer, einfacher Web-Server

- Hier ist der Quellcode (C Programmiersprache) eines (sehr) einfachen Web-Servers. Dieser Webserver liefert nur eine statische Webseite aus (diese Webseite ist fest im Quellcode des Servers eingebaut und wird nicht aus dem Dateisystem geladen):

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
```

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <err.h>

char response[] = "HTTP/1.1 200 OK\r\n"
"Content-Type: text/html; charset=UTF-8\r\n\r\n"
"<!DOCTYPE html><html><head><title>Bye-bye baby bye-bye</title>"
"<style>body { background-color: #111 }"
"h1 { font-size:4cm; text-align: center; color: black;"
" text-shadow: 0 0 2mm red}</style></head>"
  "<body><h1>Goodbye, world!</h1></body></html>\r\n";

int main()
{
  int one = 1, client_fd;
  struct sockaddr_in svr_addr, cli_addr;
  socklen_t sin_len = sizeof(cli_addr);

  int sock = socket(AF_INET, SOCK_STREAM, 0);
  if (sock < 0)
    err(1, "can't open socket");

  setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, &one, sizeof(int));
  int port = 8080;
  svr_addr.sin_family = AF_INET;
  svr_addr.sin_addr.s_addr = INADDR_ANY;
  svr_addr.sin_port = htons(port);

  if (bind(sock, (struct sockaddr *) &svr_addr, sizeof(svr_addr)) == -1) {
    close(sock);
    err(1, "Can't bind");
  }

  listen(sock, 5);
  while (1) {
    client_fd = accept(sock, (struct sockaddr *) &cli_addr, &sin_len);
    printf("got connection\n");

    if (client_fd == -1) {
      perror("Can't accept");
      continue;
    }

    write(client_fd, response, sizeof(response) - 1); /*-1:'\0'*/
    close(client_fd);
  }
}
```

- Wir erstellen ein neues Verzeichnis für unser Project und erstellen die Datei mit dem Quellcode mit Hilfe eines Text-Editors (emacs, mg, nano, vim etc)

```
mkdir ~/src
cd ~/src
$EDITOR simple-server1.c
```

- Im nächsten Schritt wird der Quellcode in eine Programm-Datei übersetzt (`simple-server`)

```
gcc -o simple-server simpler-server1.c
```

- Die Programmdatei kopieren wir in das Verzeichnis `/usr/local/bin`

```
cp simple-server /usr/local/bin
```

- Wir starten den Server im Hintergrund und testen die Funktion in dem wir uns mit einem Web-Browser an Port 8080 verbinden. Wir sollten dort eine "Hello^{H}H^{H}H^HGoodbye World" Meldung sehen. Bei jeder Verbindung gibt der Server den Text *Got connection* aus.

```
simple-server &
```

- Wenn wir uns die SELinux Label des Dienstes anzeigen lassen sehen wir das dieses Dienst `unconfined` ist, also nicht durch SELinux abgesichert

```
ps -eZ | grep simple
ls -lZ /usr/local/bin/simple-server
```

3.8.3. Initiales SELinux Policy Modul

- Wir erstellen ein neues Verzeichnis fuer das neue SELinux Policy Modul

```
mkdir ~/selinux-src
cd ~/selinux-src
```

- Der Befehl `sepolicy generate` erzeugt eine Vorlage für ein SELinux Policy Modul

```
sepolicy generate -n simple-server --init /usr/local/bin/simple-server
```

- Es werden drei SELinux Policy Quelldateien erstell

  `simple-server.te` - Type Enforcement Quellcode - auf welche Datei-Typen darf der Prozess zugreifen

  `simple-server.fc` - File Context Quellcode - welche Datei-Typen werden benutzt (und in welchen Pfaden liegen diese)

  `simple-server.if` - Interface Quellcode - Definiert die Übergänge zwischen den Dateisystem und Prozess Typen, und definiert die Regeln für die Richtlinien

  `simple-server_selinux.spec` - Quelldatei für ein RPM Paket

  `simple-server.sh` - Shell Skript zum bauen des RPM Pakets des SELinux Policy Moduls

- Diese Dateien können wir uns anschauen
- Die Policy ist im `permissive` Modus!

```
less simple-server.te
less simple-server.fc
less simple-server.if
```

- Wir testen diese SELinux Policy indem wir diese übersetzen und ein RPM-Paket erstellen. Das Paket `rpm-build` wird benötigt um ein RPM-Paket zu bauen

```
dnf -y install rpm-build
sh simple-server.sh
ls -l
```

- Die neue SELinux Policy befindet sich in der Datei `simple-server.pp`. Dieses neue SELinux Policy-Modul kann nun geladen und aktiviert werden

```
semodule -i simple-server.pp
```

- Die neue Policy wirkt sich nicht auf schon gestartete Prozesse aus. Daher stoppen wir den vorher gestarteten `simple-server` Prozess

```
pkill simple-server
```

- Wir erstellen eine SystemD Service-Unit für den Server Dienst

```
$EDITOR /etc/systemd/system/simple-server.service
```

- Die Unit-Datei

```
[Unit]
Description=a simple http server
After=syslog.target network.target

[Service]
ExecStart=/usr/local/bin/simple-server

[Install]
WantedBy=multi-user.target
```

- Die neue Systemd-Service-Datei muss mit dem richtigen SELinux Label versehen werden

```
restorecon -R -v /etc/systemd/system/simple-server.service
```

- Systemd Service-Units neu laden und den Simple-Server starten

```
systemctl daemon-reload
systemctl start simple-server
systemctl enable simple-server
systemctl status simple-server
```

- Nun den Dienst benutzen (Mit dem Web-Browser auf Port 8080 zugreifen). Das SELinux Modul ist noch im Permissive Mode, Verstösse gegen die Policy werden im Audit-Log protokolliert

```
ausearch -m avc -ts recent -c simple-server
```

- Auch das Systemd-Journal liefert Fehlermeldungen über das Programm `setroubleshoot`

```
journalctl | grep setroubleshoot
```

- Erklärungen zu den Policy-Fehlermeldungen ausgeben

```
ausearch -m avc -ts today -c simple-server | audit2why  | less
```

- Mittels des Programms `audit2allow` lassen sich Policy-Regeln aus den Audit-Meldungen erstellen. Diese Regeln sind selten 100% korrekt und müssen oft nachbearbeitet werden, helfen aber enorm bei der Erstellung eines Regelwerkes

  Der Befehl `sepolgen-ifgen` erzeugt aus den SLinux Interface-Dateien des Systems Hilfdateien für die Erstellung der Policy-Dateien

  Der Befehl `audit2allow -R` gibt die SELinux Policy-Regeln auf dem Terminal aus. Diese bauen wir per copy-n-paste in die Type-Enforcement-Quelldatei `simple-server.te` ein. Dabei muss auf die korrekte Reihenfolge der Abschnitte geachtet werden (`require` Block unter Deklarationen, `allow` Ausdrücke darunter):

```
sepolgen-ifgen -v
```

```
ausearch -m avc -ts today -c simple-server | audit2allow -R
require {
        type simple-server_t;
        class tcp_socket { bind create setopt accept listen };
}

#============= simple-server_t ==============
allow simple-server_t self:tcp_socket { bind create setopt accept listen };
corenet_tcp_bind_generic_node(simple-server_t)
corenet_tcp_bind_http_cache_port(simple-server_t)
```

- Neue Policy-Regeln in die Policy einfügen, Modul entfernen, neu kompilieren und dann neu laden

```
semodule -r simple-server
sh ./simple-server.sh
semodule -i simple-server.pp
systemctl restart simple-server
```

- Die Anwendung benutzen und testen, danach wieder das Audit-Log auf SELinux Fehler des `simple-server` Prozesses prüfen. GGf. neue Regeln erstellen und Modul und Programm neu laden
- Diese Schritte wiederholen bis keine SELinux Meldungen mehr im Audit-Log auftauchen

```
 ausearch -m avc -ts recent -c httpd -i
 <no matches>
```

- Die Anwendung ggf. für eine gewisse Zeit in Produktion im *permissive* Modus betreiben und auf SELinux Fehler prüfen
- Treten keine Fehler mehr auf, dann die Zeile `permissive simple-server_t;` in der Type-Enforcement Datei auskommentieren und das Modul im *enforcing* Modus betreiben
- Schalten wir nun das `simple-server` Modul in den *enforcing* Modus, werden wir feststellen das das Programm doch nicht wie gewünscht funktioniert
- Ein Trace des laufenden Programms mittels `strace` oder *eBPF* oder *bpftrace* zeigt das die Syscalls `shutdown` und `write` fehlschlagen. Diese werden von SELinux unterbunden, aber nicht an das Audit-Subsystem gemeldet.
- Aufgabe: Trage die Syscalls `shutdown` und `write` in die Policy ein, übersetze die Policy und teste erneut

3.8.4. Eine Änderung an einer SELinux Policy

- Unser Server lernt das Logging. Bitte den Inhalt der nachfolgenden Quelldatei-Box in die Datei `simple-server.patch` in das Verzeichnis mit dem Quellcode des `simple-server1.c` speichern:

```
--- simple-server1.c    2022-10-25 09:22:06.684216579 +0000
+++ simple-server2.c    2022-10-26 08:53:33.319698944 +0000
@@ -19,6 +19,13 @@
  int main()
  {
    int one = 1, client_fd;
+   FILE *f = fopen("/var/log/simple-server.log", "a");
+   if (f == NULL)
+   {
+       printf("Error opening file!\n");
+       exit(1);
+   }
+
    struct sockaddr_in svr_addr, cli_addr;
```

```
    socklen_t sin_len = sizeof(cli_addr);

@@ -40,7 +47,8 @@
    listen(sock, 5);
    while (1) {
      client_fd = accept(sock, (struct sockaddr *) &cli_addr, &sin_len);
-     printf("got connection\n");
+     fputs("got connection\n",f);
+     fflush(f);

      if (client_fd == -1) {
        perror("Can't accept");
```

- Nun den Patch einspielen, hier wird eine neue Datei mit dem Namen `simple-server2.c` erstellt

```
cp simple-server1.c simple-server2.c
patch -p1 simple-server2.c < simple-server.patch
```

- Das gepatchte Programm. Dieses Programm schreibt nun ein einfaches Log in die Datei `/var/log/simple-server.log`

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <err.h>

char response[] = "HTTP/1.1 200 OK\r\n"
"Content-Type: text/html; charset=UTF-8\r\n\r\n"
"<!DOCTYPE html><html><head><title>Bye-bye baby bye-bye</title>"
"<style>body { background-color: #111 }"
"h1 { font-size:4cm; text-align: center; color: black;"
" text-shadow: 0 0 2mm red}</style></head>"
  "<body><h1>Goodbye, world!</h1></body></html>\r\n";

int main()
{
  int one = 1, client_fd;
  FILE *f = fopen("/var/log/simple-server.log", "a");
  if (f == NULL)
  {
      printf("Error opening file!\n");
      exit(1);
  }

  struct sockaddr_in svr_addr, cli_addr;
  socklen_t sin_len = sizeof(cli_addr);

  int sock = socket(AF_INET, SOCK_STREAM, 0);
  if (sock < 0)
    err(1, "can't open socket");

  setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, &one, sizeof(int));
  int port = 8080;
  svr_addr.sin_family = AF_INET;
  svr_addr.sin_addr.s_addr = INADDR_ANY;
  svr_addr.sin_port = htons(port);

  if (bind(sock, (struct sockaddr *) &svr_addr, sizeof(svr_addr)) == -1) {
    close(sock);
    err(1, "Can't bind");
  }
```

```
  listen(sock, 5);
  while (1) {
    client_fd = accept(sock, (struct sockaddr *) &cli_addr, &sin_len);
    fprintf(f,"got connection\n");

    if (client_fd == -1) {
      perror("Can't accept");
      continue;
    }

    write(client_fd, response, sizeof(response) - 1); /*-1:'\0'*/
    close(client_fd);
  }
}
```

- In der SELinux-Policy-Quelldatei `simple-server.fc` definieren wir den neuen Datei-Kontext `var_log_t` für die Log-Datei

```
/var/log/simple-server.log                    gen_context(system_u:object_r:var_log_t,s0)
```

- Die SELinux Type-Enforcment Datei für `simple-server` auf *Permissive* stellen und das SELinux-Modul neu übersetzen und laden
- Den neuen Server-Dienst übersetzen, den alten `simple-server` Prozess stoppen, die neue Programm-Datei nach `/usr/local/bin` kopieren, das SELinux Label anpassen und den Dienst neu starten

```
gcc -o simple-server simple-server2.c
systemctl stop simple-server
cp simple-server /usr/local/bin
restorecon -R -v /usr/local/bin/simple-server
systemctl start simple-server
```

- Per Web-Browser die Webseite auf http://localhost:8080/ aufrufen.
- Neue SELinux Audit-Meldungen tauchen auf

```
# ausearch -m avc -ts recent
----
time->Wed Aug 24 21:17:34 2016
type=SYSCALL msg=audit(1472073454.717:1390): arch=c000003e syscall=2 success=yes exit=3 a0=400ae2
a1=441 a2=1b6 a3=21000 items=0 ppid=1 pid=7869 auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0
egid=0 sgid=0 fsgid=0 tty=(none) ses=4294967295 comm="simple-server" exe="/usr/local/bin/simple-
server" subj=system_u:system_r:simple-server_t:s0 key=(null)
type=AVC msg=audit(1472073454.717:1390): avc:  denied  { open } for  pid=7869 comm="simple-server"
path="/var/log/simple-server.log" dev="vda1" ino=268256 scontext=system_u:system_r:simple-
server_t:s0 tcontext=system_u:object_r:var_log_t:s0 tclass=file
type=AVC msg=audit(1472073454.717:1390): avc:  denied  { create } for  pid=7869 comm="simple-
server" name="simple-server.log" scontext=system_u:system_r:simple-server_t:s0
tcontext=system_u:object_r:var_log_t:s0 tclass=file
type=AVC msg=audit(1472073454.717:1390): avc:  denied  { add_name } for  pid=7869 comm="simple-
server" name="simple-server.log" scontext=system_u:system_r:simple-server_t:s0
tcontext=system_u:object_r:var_log_t:s0 tclass=dir
type=AVC msg=audit(1472073454.717:1390): avc:  denied  { write } for  pid=7869 comm="simple-server"
name="log" dev="vda1" ino=258603 scontext=system_u:system_r:simple-server_t:s0
tcontext=system_u:object_r:var_log_t:s0 tclass=dir
```

- Die Erweiterungen zur SELinux Policy ausgeben, prüfen und in die Type-Enforcement-Datei `simple-server.te` einfügen:

```
# ausearch -m avc -ts recent -c simple-server | audit2allow -R


require {
```

```
        type simple-server_t;
}

#============= simple-server_t ==============
auth_log_filetrans_login_records(simple-server_t)
logging_manage_generic_logs(simple-server_t)
```

- `simple-server` SELinux Modul entfernen, Policy neu übersetzen, Modul neu laden, testen
- Der Befehl `audit2allow` kommt hier an seine Grenzen. Es fehlen die Regeln um Log-Dateien anlegen und schreiben zu dürfen

    Diese Regeln müssen manuell in die Regel-Datei geschrieben werden

    Hierzu ist ein gutes Wissen rund um Prozess-Tracing, Syscalls und Unix-Kernel Funktionen notwendig

    Ein Blick in die bestehenden SELinux Policy Quelldateien von ähnlichen Programmen kann helfen

- Wir fügen der Type-Enforcement-Quelldatei den Typ `var_log_t` und die Klassen `file` (Syscalls `create`, `open` und `write`) und `dir` (Verzeichnis mit den Syscalls `write` und `add_name`) hinzu

```
require {
        type simple-server_t;
        type var_log_t;
        class tcp_socket { bind create setopt accept listen shutdown write };
        class file { create open write };
        class dir { write add_name };
}
```

- Wir fügen der Type-Enforcement-Quelldatei die Regeln für den Zugriff auf Dateien und Verzeichnisse im `/var/log` Verzeichnisbaum hinzu:

```
allow simple-server_t var_log_t:file { create open write };
allow simple-server_t var_log_t:dir { write add_name };
```

- Das Makro `logging_rw_generic_log_dirs`

```
    logging_rw_generic_log_dirs(simple-server_t)
```

- Solange die Policy anpassen, bis keine Permission-Meldungen im Audit-Log erscheinen
- Prüfen, das die Log-Datei korrekt erstellt wird
- Als finalen Schritt den *Permissive* Modus aus der Policy herausnehmen, Die Versionsnummer anpassen (Version 1.1.0), übersetzen und nochmals testen

```
policy_module(simple-server, 1.1.0)

########################################
#
# Declarations
#

type simple-server_t;
type simple-server_exec_t;
init_daemon_domain(simple-server_t, simple-server_exec_t)

require {
        type simple-server_t;
        type var_log_t;
        class tcp_socket { accept bind create listen setopt shutdown write };
        class file { create open write };
        class dir { write add_name };
```

```
}

#permissive simple-server_t;

########################################
#
# simple-server local policy
#
allow simple-server_t self:fifo_file rw_fifo_file_perms;
allow simple-server_t self:unix_stream_socket create_stream_socket_perms;

allow simple-server_t self:tcp_socket { accept bind create listen setopt shutdown write };
allow simple-server_t var_log_t:file { create open write };
allow simple-server_t var_log_t:dir { create write add_name };

corenet_tcp_bind_generic_node(simple-server_t)
corenet_tcp_bind_http_cache_port(simple-server_t)
domain_use_interactive_fds(simple-server_t)

files_read_etc_files(simple-server_t)

miscfiles_read_localization(simple-server_t)
logging_manage_generic_logs(simple-server_t)
logging_rw_generic_log_dirs(simple-server_t)
```

- `simple-server` SELinux Modul laden, `simple-server` Prozess per Systemd neu starten, Programm testen

CHAPTER 4. RESSOURCEN

4.1. LINUX LSM

- Loadpin Dokumentation: https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/admin-guide/LSM/LoadPin.rst
- SafeSetID Dokumentation https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/admin-guide/LSM/SafeSetID.rst
- Lockdown Links

  Lockdown as a security module

  Linux kernel lockdown, integrity, and confidentiality
- Landlock Links

  Webseite: https://landlock.io/

  Kernel Dokumentation: https://www.kernel.org/doc/html/latest/security/landlock.html
- TOMOYO

  Kernel-Dokumentation: https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/admin-guide/LSM/tomoyo.rst

  Handbuch: https://tomoyo.osdn.jp/2.6/index.html.en
- SMACK

  SMACK Kernek Dokumentation

  Heise iX: SMACK im Linux Kernel
- SELinux

  NSA SELinux https://web.archive.org/web/20201022103915/https://www.nsa.gov/what-we-do/research/selinux/

  Webseite https://selinuxproject.org/page/Main_Page
- AppArmor

  AppArmor Wiki http://wiki.apparmor.net/index.php/Main_Page

4.2. AUDIT SUBSYSTEM

- Audit Subsystem Homepage https://people.redhat.com/sgrubb/audit
- Linux Audit Mailing-Liste https://listman.redhat.com/mailman/listinfo/linux-audit
- Linux Audit Dokumentation https://github.com/linux-audit/audit-documentation/wiki
- Introduction to Linux Audit http://security-plus-data-science.blogspot.com/2017/02/introduction-to-linux-audit.html
- Audit log Normalization Part 1 http://security-plus-data-science.blogspot.com/2017/02/audit-log-normal-

ization.html

- Audit log Normalization Part 2 - CSV format http://security-plus-data-science.blogspot.com/2017/02/audit-log-normalization-part-2-csv.html
- A Linux Auditd rule set mapped to MITRE's Attack Framework https://github.com/bfuzzy/auditd-attack
- Setup and configure linux auditd https://github.com/juju4/ansible-auditd
- Best Practice Auditd Configuration https://github.com/Neo23x0/auditd/
- Splunk App for Linux Auditd https://github.com/doksu/splunk_auditd
- All-Seeing Eye or Blind Man? Understanding the Linux Kernel Auditing System https://www.sans.org/white-papers/38605/
- SUSE "Understanding Linux Audit" https://documentation.suse.com/sles/12-SP4/html/SLES-all/cha-audit-comp.html
- Fedora/Red Hat: Changes/Deprecate TCP wrappers https://fedoraproject.org/wiki/Changes/Deprecate_TCP_wrappers
- HOWTO Fedora Enable Auditing https://github.com/linux-audit/audit-documentation/wiki/HOWTO-Fedora-Enable-Auditing

4.3. SELINUX

- SELinux FOR RED HAT DEVELOPERS https://www.redhat.com/en/files/resources/en-rhel-selinux-devlopers-120114.pdf
- Fedora Security-Enhanced Linux User Guide https://docs.fedoraproject.org/en-US/Fedora/13/html/Security-Enhanced_Linux/index.html
- Fedora SELinux FAQ - Frequently-asked questions about Security Enhanced Linux https://docs.fedoraproject.org/en-US/Fedora/13/html/SELinux_FAQ/index.html
- A Brief Tour of Linux Security Modules https://www.starlab.io/blog/a-brief-tour-of-linux-security-modules/
- SELinux Struggles with BIND Startup https://www.isc.org/blogs/selinux-struggles-bind/
- `named_selinux` Manual page https://linux.die.net/man/8/named_selinux (the man page on your system is likely more up-to-date)
- Policy Quellcode Beispiele https://github.com/SELinuxProject/refpolicy

4.4. ANDERES

- Instrumenting BIND 9 on Linux with BCC/eBPF https://www.youtube.com/watch?v=VYpZg89NJeA