Mérési útmutató az „Access Control (ACC)” című méréshez

2017. március 10.

A mérést kidolgozta:
Papp Dorottya
Lestyán Szilvia

BME, CrySyS Adat- és Rendszerbiztonság Laboratórium

Tartalomjegyzék

1. Bevezetés

A mérés célja különböző rendszereken megvalósított hozzáférésvédelmi megoldások vizsgálata. Ennek kapcsán a hallgatók megismerkednek a hagyományos UNIX hozzáférés-védelmi mechanizmusaival: `suid` bit, `sgid` bit. A modern hozzáférés-védelmi mechanizmusok közül a mérésen szerepelnek a hozzáférés-védelmi listák és a SELinux alapjai.

2. Hagyományos UNIX hozzáférés-védelem

Többfelhasználós rendszerekben alapvető fontosságú, hogy az egyes felhasználók saját adatait a lehetőség szerint megvédjük a szándékos vagy szándékolatlan rongálástól, nem is beszélve a magántitok védelméről és más kérdésekről. E célok biztosítására a UNIX egy tulajdonosi kategóriák és tevékenységek szerinti engedélyezési rendszert használ, s ennek alapján dönt az állományokhoz való hozzáférés engedélyezéséről, illetve megtagadásáról.

Egy UNIX felhasználót a felhasználói azonosítója (uid) azonosít a rendszerben, ezenfelül minden felhasználó valamilyen csoportba is tartozik, így van egy csoportazonosítója (gid) is. Amikor valamilyen módon hozzá szeretnénk férni egy fájlhoz vagy mappahoz, a rendszer aszerint sorol be minket, hogy milyen tulajdonosi viszonyban vagyunk az adott fájllal: lehetünk tulajdonosa (azonos uid), lehetünk csoporttagok (azonos gid), vagy "egyéb". A tulajdonost a `chown`, a csoportot pedig a `chgrp` paranccsal konfigurálhatjuk.

A fájllal kapcsolatos tevékenységek szempontjából három fő csoport van: az állomány olvasása, írása, illetve végrehajtása (mappa esetében keresése). Egy-egy fájlművelet elvégzése előtt a UNIX ellenőrzi, hogy melyik tulajdonosi kategóriába esünk, s utána azt, hogy ebben a kategóriában engedélyezett-e vagy sem a végrehajtani kívánt művelet.

A hozzáférési jogosultságok az `ls -la` lista alapján szemléltethetőek a legkönnyebben:

```
total 364
-rw-r--r-- 1 demo guest 18 Aug 23 20:42 file1
drwxrwxrwx 3 demo guest 1024 Aug 23 20:59 newdir
-rw-rw-rw- 1 demo guest 18 Aug 23 20:42 newfile
lrwxrwxrwx 1 demo guest 10 Mar  9 20:32 sample -> text
-rw-rw-rw- 2 demo guest 18 Aug 23 20:59 text
```

A lista baloldali oszlopa tartalmazza a hozzáférési jogosultságokat. Az oszlop legszélső mezője a fájltípust kódolja:

- közönséges fájl (`-`)
- mappa (`d`)
- speciális pipe (`p`)

- szimbolikus link (`l`)
- karakteres készülékmeghajtó (device driver, `c`)
- blokkos készülékmeghajtó (`b`)

A fájltípus után következő kilenc karakter kódolja a jogosultsági biteket háromszor hármas bontásban: olvasás (`r`), írás (`w`) és végrehajtás (`x`). Ha egy művelet engedélyezett, a neki megfelelő betű látszik a listán, ha nem, a `-` karakter jelzi a tiltást. Előfordulhat, hogy más karakterek vannak ebben a kilenc karakterben, néhányat a továbbiakban ismertetünk. Az első hármas csoport a tulajdonos, a második a csoport, végül a harmadik a többiek jogosultságait mutatja. Konkrét példát véve, a fenti listán szereplő `file1` egy közönséges fájl, amit a tulajdonosa írhat és olvashat, de nem hajthat végre (`r--`), ám a csoporttagok és a többiek csak olvashatják.

A jogosultsági biteknek nem csak karaktereket, hanem számokat is megfeleltethetünk. A jogosultsági bitek egymás mellett megfelelnek egy kettes számrendszerbeli számnak, amit egy oktális számmal kódolhatunk. A kettes számrendszerbeli reprezentáció leírásánál az `r` bit áll a legnagyobb helyiértékű helyen, míg az `x` bit a legkisebb helyiértékű. Adott helyiértéken akkor szerepel 1-es, ha az adott bit be van billetve engedélyezésbe. Ha nincs bebillentve, akkor az adott helyiértéken 0 szerepel. Például, az `rw-` jogosultság megfelel a kettes számrendszerbeli 110-nak, amit oktálisan 6-ként kódolunk.

Attól függően, hogy közönséges fájlról vagy mappáról van szó, az olvasás-írás-végrehajtás fogalma változik. Közönséges fájloknál az olvasás a fájl megnyitását és tartalmának kiírását jelenti. Az írás a tartalom módosítása, a végrehajtás pedig a fájl futtatása végrehajtható állományként. Mappáknál az olvasási jogosultság a mappa tartalmának listázására vonatkozik. Az írási jogosultsággal rendelkezők módosíthatják a mappa tartalmát: fájlokat hozhatnak létre, törölhetnek, átnevezhetnek.

Ezzel szemben egy mappa olvasása azt jelenti, hogy tudhatunk a létezéséről, listázni tudjuk magát a mappát és a benne található fájlok neveit (pl. `ls` paranccsal). Egy mappa akkor írható, ha bejegyzéseket tudunk létrehozni, módosítani, vagy törölni benne, illetve törölhetjük/létrehozhatjuk magát a mappát (kapcsolódó parancsok: `mkdir`, `rmdir`, `mv`, `cp`, `rm`, `ln`). Mappák esetén a végrehajtási jogosultság megfelel a belépés (traverse) engedélynek, enélkül nem érhetjük el a mappa tartalmát, még akkor sem, ha azt listázni tudjuk. Például tegyük fel, hogy van egy `prog` nevű végrehajtható állományunk, ami egy olyan mappában szerepel, amelyikre engedélyezett az olvasás, de nem engedélyezett az írás és a végrehajtás (`r--`). Ekkor magát a mappát látjuk ugyan a kilistázáskor, és azt is megtudhatjuk, hogy tartalmazza a `prog` nevű állományt, de az állomány tartalmát és metaadatait (méret, jogosultságok, utolsó hozzáférés ideje, stb) nem érhetjük el. Ha ugyanez a mappa nem olvasható, de kereshető (`--x`), akkor nem fogjuk tudni kilistázni

az benne található `prog` végrehajtható állományt, de ha kiadjuk a végrehajtási parancsot, akkor az állomány végre fog hajtódni, hiszen be tudtunk lépni a mappába, hogy elérjük.

3. A `suid` és `sgid` bitek, valamint a sticky bit

A UNIX típusú rendszerekben a programok azokkal a felhasználói jogokkal futnak, amivel az őket elindító felhasználó rendelkezik. Bizonyos esetekben azonban ez nem szerencsés. Talán a legegyszerűbb példa a felhasználói azonosító megváltoztatására az accounthoz tartozó jelszó megváltoztatása. Egy különleges privilégiumokkal nem rendelkező felhasználó általában nem írhatja közvetlenül a rendszer jelszófájlját, hiszen akkor bármikor korlátlan jogokhoz juthatna. Mégis, szükség van arra, hogy a saját jelszavát megváltoztassa, amihez viszont írnia kell a jelszófájlt. Ezt az ellentmondást oldják fel a suid és az sgid bitekkel.

A suid (Set User Identification) a felhasználói azonosító megváltoztatása, ezáltal a privilégiumok megváltoztatása.A suid bitet a jogosultságok között a tulajdonosra vonatkozó végrehajtási bit helyén láthatjuk, `s` jelzéssel:

```
-rwsr-xr-x 1 root root 28896 Sep 7 13:40 /usr/bin/passwd
```

Az `s` a set-uid (set user-id) jog engedélyezését jelzi, ha a tulajdonosi kategóriánál szerepel. Ha e jog engedélyezett, akkor akárki is futtatja a szóbanforgó programot, a futtatás idejére olyan jogokkal fog rendelkezni, mint a program tulajdonosa. A példában szereplő `passwd` program esetében ez a rendszergazdát jelenti.

A UNIX újabb verziói lehetőséget biztosítanak arra is, hogy egy felhasználó több csoportba is tartozhasson (másodlagos csoportok). Konfigurálásukhoz rendszergazdai jogosultságra van szükség. A másodlagos csoportok a csoportmunkák támogatásánál előnyösek. Így lehetőség nyílik arra, hogy valaki elsődlegesen a saját felhasználójának csoportjába tartozzék, de minden olyan fájlhoz hozzáférjen a másodlagos csoporton keresztül, amely fájlok olyan projekthez tartoznak, amelyben ő résztvesz. Ezt úgy érjük el, hogy a projekt fájlainak csoport tulajdonosa az a csoport, amit a csapattagok másodlagos csoportként használnak.

A `sgid` bit (Set Group Identification) a csoportazonosító megváltoztatására szolgál, beállítása esetén a program annak a csoportnak a jogaival fog futni, amelyik csoportnak a fájl a birtokában van. A sgid illetve a set-gid (set group-id) jogot a tulajdonos csoport jogosultsági bitjei között láthatjuk a végrehajtás helyén `s` karakterrel jelölve. A `sgid` bitet mappák esetén is be lehet kapcsolni. Ennek eredményeként, ha ebben a könyvtárban bárki létrehoz egy fájlt (ehhez a többi jognak rendben kell lennie), akkor a fájl csoporttulajdonosa nem az a csoport lesz, amelyikbe a felhasználó tartozik, hanem az, akinek a könyvtár a birtokában van.

A `sticky bit` bekapcsolása fájlok esetén azt jelzi az operációs rendszernek, hogy a fájlt tartsa a memóriában a végrehajtás után is. Ennek a tulajdonságnak akkor van értelme, ha azt szeretnénk, hogy egy program minél gyorsabban induljon el, ne kelljen várni a betöltődésére. A sticky bitet is be lehet kapcsolni mappák esetén is. Az ilyen bittel ellátott mappákba bárki írhat fájlokat (feltéve, hogy erre jogosultsága van), de mindenki csak a sajátját törölheti. Ezt a lehetőséget azért tervezték, hogy az olyan, mindenki által írható könyvtárakban, mint például a `/tmp`, a felhasználók ne tudják a másik felhasználó által írt fájlokat módosítani, letörölni.

4. Hozzáférés-védelmi listák

A hagyományos UNIX hozzáférés-védelem egyik hiányossága, hogy a hozzáférést csak korlátozott részletességgel konfigurálhatjuk be. A hagyományos keretek között, ha egy fájlhoz másik felhasználónak is engedélyt akarunk adni, akkor hozzá kell adnunk őt a fájlt birtokló csoporthoz. Ekkor azonban minden olyan fájlhoz hozzáférést kapott a kérdéses felhasználó, amit ez a csoport birtokol, pedig nem ezt szerettük volna elérni. Az Access Control List (ACL) erre a problémára nyújt megoldást úgy, hogy a hagyományos hozzáférési listát kiegészíthetjük más felhasználókra és csoportokra vonatkozó engedélyekkel, valamint új maszkkal. Az ACL használatával a fent említett probléma kiküszöbölhető azzal, hogy a fájlhoz tartozó ACL-be felvesszük a kérdéses felhasználót olvasási jogosultsággal. Ekkor a rendszer a felhasználót nem a másoknak adott jogosultságok alapján bírálja el, hanem azok alapján a jogosultságok alapján, amit az ACL rá vonatkozóan ír.

Az ACL-ek használatához először telepíteni kell az `acl` csomagot (ha nincs telepítve automatikusan), majd a fájlrendszert az `acl` engedélyezésével kell mountolni. A csomag többek között telepíti a `setfacl` és a `getfacl` parancsokat, ezekkel tudjuk konfigurálni és áttekinteni az ACL-eket. A mountoláshoz a `/etc/fstab` fájlt kell szerkeszteni és a megfelelő bejegyzéshez felvinni az `acl` opciót. Például:

```
UUID=<UUID> /home ext4 defaults,acl 0 2
```

A opció megadása után a partíciót újra kell mountolni (`umount`, majd `mount`, esetleg `reboot`).

Amennyiben létezik ACL egy objektumra, a listázáskor a hagyományos jogosultságok mellett egy `+`-t fogunk látni. A mappákra és fájlra definiált ACL-eket a `getfacl` paranccsal nézhetjük meg:

```
$ getfacl teamfolder
# file: teamfolder
# owner: user
# group: team
user::rwx
```

```
user:angela:rw-
group::rwx
other::r-x
default:user:rw-
default:group:rw-
default:other:r--
```

A kimenetben láthatjuk az objektum nevét, amire az ACL vonatkozik, valamint az objektum tulajdonos felhasználóját és csoportját. Az ACL tartalmazza, hogy a tulajdonos felhasználónak, a csoportnak és másoknak milyen jogosultságai vannak.

Amennyiben egy ACL csak a tulajdonos felhasználóra, csoportra és másokra tartalmaz bejegyzéseket, *minimális* ACL-nek nevezzük. Amennyiben `default` és/vagy néven nevezett felhasználókra/csoportokra vonatkozóan is tartalmaz bejegyzéseket, *kiterjesztett* ACL-nek nevezzük. A `default` bejegyzések rögzítik, hogy az újonnan létrehozott fájlok milyen jogosultságokat örököljenek a fájlrendszer-hierarchiában felettük álló objektumtól.

Az ACL-ek segítségével a fájlok eléréséhez maszkot is definiálhatunk. Hagyományos esetben a *maszk* a fájlok és mappák létrehozásakor kap szerepet, a default értéket az `umask` paranccsal nézhetjük meg. Ez egy oktális szám, amelyet a default engedélyek oktális értékéből von le a rendszer. Létrehozáskor a fájl default engedélyei a 666 értéknek felelnek meg (mappánál 777), ebből vonódik le a default maszk érték (általában 022). Végeredményként fájlok esetén általában 644-et kapunk, vagyis `rw-r--r--` engedélyek lesznek a fájlon. Ez a mechanizmus általában jó hozzáférés védelmet eredményet, de minden felhasználó csak egy maszk értékkel rendelkezik, vagyis minden általa létrehozott fájl ugyanazokkal a jogosultságokkal fog létrejönni. Ezek a jogosultságok azonban nem minden esetben vannak összhangban az adott mappában érvényes fájl jogosultsági irányelvvel, elképzelhető, hogy olyan jogosultságokat is tartalmaznak, amit az irányelv tilt. Ilyen esetekben az ACL segítségével megadhatunk egy lokálisan értelmezett maszk értéket, ami leírja, hogy mik a maximálisan engedélyezhető jogosultságok. Ha például az ACL maszk szerint maximum `rw-` jogosultság engedélyezett a fájlokon, akkor hiába látjuk egy végrehajtható állománynál az `r-x` jogosultságot, a maszk miatt nem fogjuk tudni futtatni.

5. Apache2 webszerver

A mérés során az Apache2 webszervert (`https://httpd.apache.org/`) fogjuk felhasználni a SELinux képességeinek bemutatásához. Ezért most egy rövid összefoglaló következik a webszerver konfigurálásáról.

Az Apache2 konfigurációja ún. direktívák elhelyezésével történik szöveges konfigurációs fájlokban. A konfigurációs fájlok minden sorában egy-egy

direktíva található, melyhez az argumentumokat whitespace karakterekkel elválasztva adhatjuk meg. A # jellel kezdődő sorok kommentek.

Hogy több weboldalt is üzemeltethessünk ugyanazon a szerveren, virtuális hoszt konfigurációkat kell létrehoznunk a `VirtualHost` direktívával. Argumentumként az IP-címet és portot kell megadnunk, a `*` minden címre illeszkedik. A `VirtualHost` direktíván belül többféle konfigurációs lehetőségünk van. Az alábbi felsorolás nem teljeskörű, csak a mérés elvégzéséhez feltétlen szükséges direktívákat tartalmazza:

- `ServerName`: weboldal neve
- `DocumentRoot`: a mappa, ahol a weboldal fájljai találhatóak
- `Directory`: kliensek jogosultságai meghatározása adott mappára
- `ErrorLog`: a fájl vagy program, ahova az apache a weboldalhoz tartozó hibákat logolja
- `AccessLog`: a fájl vagy program, ahova az apache a beérkező kéréseket logolja

A konfigurált weboldalakat engedélyezni kell az apache számára, majd újra be kell tölteni az apache2 konfigurációt.

6. Security Enhanced Linux

A Security Enhanced Linux (SELinux) egy biztonsági kiegészítés a Linux kernelhez, mely nagyobb felügyeletet biztosít felhasználók és rendszeradminisztrátorok számára egyaránt azáltal, hogy *mandatory access controlt* (MAC) implementál. Ez azt jelenti, hogy képes korlátozni azt is, hogy az erőforrás tulajdonosa hogyan férhet hozzá az erőforráshoz. Ezt a hagyományos hozzáférés védelmi mechanizmusok nem tudták megtenni, mivel azok a *discretionary access control* (DAC) implementációi. DAC esetén sikeres hitelesítés után a tulajdonos döntheti el az adott erőforráshoz tartozó hozzáférési jogosultságokat, pl. bármikor átadhatja az erőforrást másnak, vagy információt oszthat meg az erőforrásról.

A SELinux három működési móddal rendelkezik:

- *Enforcing:* a SELinux megakadályoz minden tevékenységet, kivéve azokat, amelyek explicit engedélyezünk; a tevékenységeket naplózza
- *Permissive:* a SELinux engedélyezve van és aktívan naplóz, de nem lép közbe, ha tiltania kéne
- *Disabled:* a SELinux nincs engedélyezve a gépen, sem logolás, sem tiltás nem történik

A aktuális működési módot a `getenforce` vagy `sestatus` parancsokkal tudjuk megnézni és a `setenforce` paranccsal vagy a `/etc/selinux/config` fájlon keresztül tudjuk beállítani.

6.1. Alapfogalmak

A hozzáférés-védelem érvényre jutattásához minden fájl, socket, felhasználó, folyamat, stb. rendelkezik egy címkével, amit SELinux *kontextusnak* nevezünk. A kontextus formátuma `user:role:type:level(optional)`. A fájlok és mappák kontextusait a fájlrendszeren tárolja a számítógép, a folyamatok, portok, stb. kontextusát pedig a kernel menedzseli. A kontextusok menedzselését *labelingnek* vagy címkézésnek is hívjuk. A kontextusok megtekintéséhez a `-Z` kapcsolót használhatjuk a különböző listázó programoknál, pl. `ls -Z`.

A kontextus `type` része egyfajta csoportosítást biztosít a különböző fájlok és objektumok között, folyamat esetében *domainnek* nevezzük. A *type enforcement* mechanizmus felelős annak menedzseléséért, hogy mely kontextussal rendelkező objektumok között megengedett az interakció. Például, az Apache2 webszerver, melyek domainje `httpd_t`, használhatja a weboldalak fájljait, amelyek típusa `httpd_sys_content_t`, de nem érheti el a `/etc/shadow` fájlt, melynek típusa `shadow_t`. Természetesen előfordulhat, hogy egy folyamatnak más típussal rendelkező objektumhoz kell hozzáférnie. Ezt a kontextusok közötti átmenetet *domain transionnek* nevezzük és külön engedélyezni kell.

A kontextus `user` része adja meg a SELinux felhasználót, amely *nem egyezik meg a hagyományos Linux felhasználókkal.* A SELinux felhasználó az a személy, aki a hozzáférés védelmi szabályok szerint bizonyos szerepeket tölthet be. A szabályok minden Linux és SELinux felhasználó között 1:1 megfeleltetést biztosítanak, így a Linux felhasználók megöröklik a SELinux felhasználók korlátozásait. A megfeleltetéseket a `semanage login -l` paranccsal nézhetjük meg:

```
# semanage login -l
Login Name      SELinux User      MLS/MCS Range        Service
__default__     unconfined_u     s0-s0:c0.c1023       *
root            unconfined_u     s0-s0:c0.c1023       *
system_u        system_u         s0-s0:c0.c1023       *
```

Amennyiben egyik felhasználónak se felel meg a Linux felhasználó, a táblázat `__default__` sora jut érvényre. A 1. táblázat egy rövid összefoglalást ad néhány fontosabb SELinux felhasználóról.

SELinux user	Leírás
unconfined_u	Szinte semmilyen korlátozás
root	SELinux felhasználó a `root` számára
sysadm_u	SELinux felhasználó rendszergazdai jogosultságokkal
staff_u	SELinux felhasználó rendszergazdai és végfelhasználói jogosultságokkal egyaránt
user_u	SELinux felhasználó különleges jogosultságokkal nem rendelkező felhasználók számára
system_u	Speciális SELinux felhasználó a rendszerfolyamatoknak

1. táblázat. SELinux felhasználók

A kontextus `role` része jelzi a SELinux felhasználó által betöltött szerepet. A SELinux szerepek határozzák meg, hogy milyen domainben lehet a SELinux felhasználó, és ezáltal mely objektumokkal léphet interakcióba. A SELinux felhasználók által felvehető szerepeket a `semanage user -l` paranccsal nézhetjük meg:

```
# semanage user -l

                Labeling
SELinux User    Prefix     SELinux Roles

guest_u         user       guest_r
root            user       staff_r sysadm_r system_r unconfined_r
staff_u         user       staff_r sysadm_r system_r unconfined_r
sysadm_u        user       sysadm_r
system_u        user       system_r unconfined_r
unconfined_u    user       system_r unconfined_r
user_u          user       user_r
xguest_u        user       xguest_r
```

A táblázat több oszlopot is tartalmaz, azonban milyen ezek nem képezik a mérés anyagát, a példában sem szerepelnek.

A hozzáférés védelemet meghatározó szabályok alkotják *policy*-t, melynek felépítése moduláris, több bináris fájlban tárolhatóak a szabályok. Funkcionalitása szerint a policy-t az alábbi csoportokba oszthatjuk:

- *Minimum:* néhány daemon folyamatot korlátoz, minden más objektum a számítógépen a tradícionális hozzáférés védelem alá tartozik
- *Targeted:* számos daemon folyamathoz definiál korlátozásokat, felhasználókat és más objektumokat is képes korlátozni

- *Multi-level security:* többszintű hozzáférés-védelem, nem szerepel a mérés anyagában
- *Refpolicy:* meghatározza, hogy a policy-t buildelő folyamatra milyen korlátozások érvényesek

A aktuális policy besorolását a `/etc/selinux/config` fájlban állíthatjuk be. Minden olyan tevékenység, amit a policy szerint meg kell (permissive módban kéne) akadályoznia a SELinuxnak, naplózásra kerül. CentOS-en a naplóbejegyzések gyűjtésére a `/var/log/audit/audit.log` fájlt használjuk. A SELinuxhoz kapcsolódó naplóbejegyzések az "AVC" kulcsszóval vannak ellátva, alább láthatunk egy példát. Mivel a naplóbejegyzések formátuma nem közérthető, érdemes őket átadni az `audit2why` segédprogramnak, ami rövid értelmező szöveget fűz a bejegyzésekhez.

```
type=AVC msg=audit(1489091753.306:78): avc:  denied  { append }
for  pid=981 comm="httpd" name="error.log"
dev="dm-0" ino=67147261
scontext=system_u:system_r:httpd_t:s0
tcontext=system_u:object_r:default_t:s0
tclass=file

        Was caused by:
                Missing type enforcement (TE) allow rule.

                You can use audit2allow to generate a loadable
                module to allow this access.
```

A fenti naplóbejegyzés egy SELinux által dobott hibaüzenetet örökít meg. A bejegyzés szerint a SELinux megakadályozta a `httpd` programnak, ami a `system_u:system_r:httpd_t:s0` kontextusban futott, hogy hozzáférjen a `system_u:object_r:default_t:s0` kontextussal ellátott `error.log` objektumhoz.

A telepített policy-hez mi is adhatunk hozzá újabb szabályokat. Ezt manuálisan, vagy az `audit2allow` segédprogrammal tehetjük meg. A segédprogramot használatához az `/var/log/audit/audit.log` fájlból a megfelelő SELinux hibaüzeneteket pipe-on keresztül adhatjuk át az `audit2allow`-nak, ami elkészíti a szabályhalmaz leírását. Alább láthatunk egy példa szabályhalmazt:

```
module sudo-resources-policy 1.0;

require {
        type sysadm_t;
        type devpts_t;
        type sysadm_sudo_t;
```

```
        type init_t;
        type user_tmpfs_t;
        type tmpfs_t;
        class fd use;
        class unix_dgram_socket sendto;
        class dir search;
        class file { read getattr open };
        class chr_file { read write getattr ioctl };
}

#============= sysadm_sudo_t ==============
allow sysadm_sudo_t devpts_t:chr_file { read write getattr ioctl };
allow sysadm_sudo_t init_t:fd use;
allow sysadm_sudo_t sysadm_t:unix_dgram_socket sendto;
allow sysadm_sudo_t tmpfs_t:dir search;
allow sysadm_sudo_t user_tmpfs_t:dir search;
allow sysadm_sudo_t user_tmpfs_t:file { read getattr open };
```

A szabályhalmaz három részből áll. Az első rész a `module` parancs, ami azonosítja a szabályhalmaz nevét és verzióját. A szabályhalmaz nevének mindig egyedinek kell lennie. A második rész a `require` blokk, ami leírja, hogy milyen típusoknak, osztályoknak és szerepeknek kell szerepelniük a policy-ben, mielőtt ezt a szabályhalmazt telepíthetjük. Az utolsó részben (`allow` parancsok) vannak felsorolva az engedélyezési szabályok.

Az `audit2allow` kimeneteként két fájl keletkezik: `<policy_neve>.pp` és `<nev>.te`. Az előbbi a szabályhalmaz bináris leírása, amit `semodule -i <nev>.pp` paranccsal adhatunk a policy-hez, a másik szövegesen tartalmazza a szabályokatKimenetként két fájl keletkezik: `<policy_neve>.pp` és `<nev>.te`. Az előbbi a szabályhalmaz bináris leírása, amit `semodule -i <nev>.pp` paranccsal adhatunk a policy-hez, a másik szövegesen tartalmazza a szabályokat.

Megjegyzés: Egyes Linux operációs rendszerek esetében a fejlesztők nem tartják karban a SELinux policy-ket, ilyen például a Debian és az Ubuntu is. Ha a való életben SELinuxot szeretnénk használni, érdemes egy olyan Linux disztribúciót használni, ahol a letölthető policy-ket a fejlesztő csapat aktívan karban tartja (pl. Red Hat, CentOS).

Egyszerűbb szabályokat a `semanage` parancs segítségével is konfigurálhatunk. Ha pl. meg akarjuk változtatni egy fájl vagy mappa default kontextusának típus mezőjét, azt a következő paranccsal tehetjük meg:

```
semanage fcontext -a -t <type> "<path>"
```

A `<type>` az a típus, amire a default típust változtatni akarjuk. A rendszeren létező kontextusokat és így a típusokat is a *semanage fcontext -l* paranccsal

nézhetjük meg. A parancsnak hosszú a kimenete, érdemes azt elmenteni egy fájlba. A `<path>` a megváltoztatni kívánt objektum(ok) elérési útvonala. Az útvonal lehet konkrét (pl. `/etc/passwd`), de lehet egy reguláris kifejezés is, ekkor a SELinux minden, a reguláris kifejezés által megadott útvonalra illeszkedő objektumra a megadott címkét fogja beállítani. Ha változtattunk a címkézési szabályokon, a változtatásokat a `restorecon` paranccsal tudjuk érvényre juttatni.

Megjegyzés: A `semanage fcontext` paranccsal létrehozott szabályok permanensen tárolódnak. Ha csak a következő újraindításig szeretnénk, hogy egy szabály érvényes legyen, a `chcon` parancsot érdemes használni.

Léteznek olyan szabályhalmazok a targeted policy-ban, amiknél egyes rendszereken az engedélyezés a logikus, másoknál a tiltás. Ezeket a szabályhalmazokat *booleannak* nevezzük. Ilyen például a `httpd_enable_cgi`, ami a CGI szkriptek futtathatóságát szabályozza. Ha webszevert üzemeltetünk és a weboldalon vannak CGI szkriptek, akkor engedélyezni kell ezt a booleant, ha nem webszerver a számítógépünk, vagy nincsenek CGI szkriptek, akkor tiltani. Az elérhető booleanokat a `getsebool` paranccsal nézhetjük meg, értéküket pedig a `setsebool <boolean> <on/off>` paranccsal állíthatjuk.

7. Feladatok

A mérés során egy webszervert fogunk felkonfigurálni, ami egy megosztott mappában tárolt weboldalt hosztol. A megosztott mappában egy képzeletbeli vállalat dolgozói tárolnak dokumentációkat és egy wiki oldalt. A vállalatnak szakmai gyakorlatos dolgozói is vannak, akik csak időlegesen kapnak hozzáférést a rendszer egy-egy erőforrásához.

7.1. Megosztott mappa konfigurálása

7.1.1. Ismerkedés a rendszerrel

Kapcsolódjon a virtuális géphez a Remmina programon keresztül! A bejelentkezéshez használja a `user`/`password` párost!

Milyen felhasználói fiókok vannak a gépen és melyik csoportokhoz tartoznak a különböző felhasználók? Csak azokat a felhasználói fiókokat sorolja fel, amelyekhez parancssoron be lehet jelentkezni!

Melyik felhasználó tud rendszergazdai jogosultságot igénylő parancsokat futtatni?

7.1.2. Megosztott mappa létrehozása

Váljon `root`tá és hozza létre a `/data/docs` mappát! Ezt a mappát fogják használni a felhasználók a dokumentációk és a wiki oldal létrehozásához.

A mappa tulajdonos felhasználója legyen `eric`, tulajdonos csoportja pedig a `contentdevs` csoport!

7.1.3. Mappa megosztása csoporton belül

Váltson át az `eric` felhasználói fiókra (jelszó: `eric`) és hozzon létre egy üres szöveges fájlt a `/data/docs` mappában! Milyen jogosultságokkal jött létre a fájl? Ki(k) a fájl tulajdonosai?

Az `alice` nevű felhasználó szintén tagja a `contentdevs` csoportnak. Milyen jogosultsága van a létrejött fájlhoz? Válaszát indokolja és parancssorban ellenőrizze (`alice` jelszava `alice`)!

Tud-e új fájlt létrehozni `alice` a `/data/docs` mappában? Válaszát indokolja és ellenőrizze a parancssoron!

Adjon írási jogosultságot a tulajdonos csoportnak a `/data/docs` mappán!

Tudja-e `alice` módosítani az `eric` által látrehozott fájlt? Válaszát indokolja és ellenőrizze a parancssoron!

Hogy orvosolná a problémát? Több megoldást is vázoljon!

Billentse be a `sgid` jogosultsági bitet a `/data/docs` mappán és próbálkozzon újra! Mi változott? Tudná-e `eric` módosítani az `alice` által létrehozott fájlt, vagyis valóban megosztott-e a mappa a `contentdevs` csoport tagjai között?

7.1.4. Hozzáférés-védelmi lista konfigurálása

A megosztott tartalmakért felelős csoporthoz beosztanak egy szakmai gyakorlatos hallgató (`joe`). A hallgató feladata segíteni a tartalom-fejlesztésben és a wiki oldal elkészítésében. A rendszergazda azonban nem szeretné `joe`-t hozzáadni a `contentdevs` csoporthoz. Miért?

Ellenőrizze, hogy a rendszeren telepítve van-e az `acl` csomag!

Konfigurálja a virtuális gépet úgy, hogy a fő partíción engedélyezve legyenek az ACL-ek!

Konfigurálja a `/data/docs` mappa hozzáférés védelmi listáját úgy, hogy annak mostani és jövőbeli tartalmához is hozzáférjen `joe`! Megoldását ellenőrizze (`joe` jelszava `joe`)! Hogyan néz ki a módosított ACL?

7.2. Wiki oldal létrehozása

7.2.1. A weboldal elkészítése

A tartalomfejlesztésért felelős csoport úgy döntött, hogy a wikihez tartozó weboldal fájljait a `/data/docs/wiki_html/` mappában fogják tárolni. Hozza létre ezt a mappát, benne egy minta weboldallal!

A virtuális gépre előre telepítettük az apache2-t. Ellenőrizze, hogy a webszerver működik és képes GET kéréseket kiszolgálni!

Ahhoz, hogy megértsük a SELinux type enforcement mechanizmusát, először SELinux nélkül fogjuk elkészíteni a wiki oldalt és csak utána kapcsoljuk be a SELinuxot. Állítsa a SELinuxot permissive módba!

CentOS esetén a apache2-höz tartozó konfigurációs fájlok a `/etc/httpd` mappában vannak. Nyissa meg a `/etc/httpd/conf/httpd.conf` fájlt! Hova kell elhelyezni a weboldalak kiszolgálásához szükséges virtuális hosztok konfigurációs fájljait?

Hozzon létre egy virtuális hosztot, ami a `www.wiki.company.com` oldalt szolgálja ki a `/data/docs/wiki_html/` mappából!

Mivel a megadott `www.wiki.company.com` címet nem regisztráljuk a DNS rendszerben, a virtuális gépen állítsa be az `/etc/hosts` fájlban, a `www.wiki.company.com` weboldalt a virtuális gép 127.0.0.1-ként oldja fel!

Indítsa újra az apache2-t és ellenőrizze, hogy elérhető a minta wiki oldal!

7.2.2. A SELinux type enforcement mechanizmusa

Állítsa a SELinuxot enforcing módba és próbálja újra elérni a weboldalt! Mit tapasztal?

Hogy jobban megértsük, mi történt, telepítsük a `policycoreutils-python` csomagot, amit tartalmazza az `audit2why` segédprogramot!

Keresse ki a `/var/log/audit/audit.log`-ból a SELinuxhoz tartozó bejegyzéseket az apache2 futásáról! Hogyan néznek ki a bejegyzések?

Adja át a kikeresett bejegyzéseket az `audit2why` segédprogramnak! (Amennyiben hibaüzenetet kap, miszerint hiányzik az `/etc/selinux/targeted/contexts/files/file_contexts.local`, hozza létre a fájlt manuálisan!) Mit javasol a segédprogram, hogyan szüntesse meg a problémát? Egyetért a segédprogrammal? Válaszát indokolja!

A SELinux többféle típust is definiál a webszerver számára elérhető fájlok címkézéséhez attól függően, hogy milyen szerepet tölt be a fájl és azt hogyan használja a webszerver. A `semanage fcontext` parancs megfelelő paraméterezésével nézze meg a `httpd`-hez tartozó kontextusokat és soroljon felbelőlük néhányat! Melyik kontextus való csak olvasható weboldalak forrásfájljaihoz és melyik a webszerver által használt naplófájlokhoz?

A `semanage fcontext` parancs megfelelő paraméterezésével vegyen föl szabályt, amely `httpd_sys_content_t` típust állít be a wiki oldal forrásfájlainak és `httpd_log_t` a naplófájloknak! Utána címkézze újra a `/data` mappát!

Ellenőrizze, hogy a típusok átállítása megtörtént és próbálja meg újra elérni a wiki oldalt! Mit tapasztal?