Hálózatkezelés NSA Security-Enhanced Linux alatt

Egy gyakorlati példa segítségével küzdjük le az SELinux bonyolultsága miatti félelmeinket, és egyszerűbb kiszolgálóinkat is vértezzük fel SELinux alapú védelemmel.

Írásomban áttekintem, az ***SELinux*** segítségével hogyan növelhető a hálózati rendszerek biztonsága, valamint ismertetem hálózati vonatkozású védelmi vezérlőinek felépítését és megvalósítását. Ezután átveszünk egy az ***SELinux*** házirendjének egy egyszerű hálózati alkalmazás zárolására való használatát szemléltető példát.

Az SELinux szerepeinek, típusainak és tartományainak áttekintése

Az ***SELinux*** erőteljes általános védelmet nyújt a hálózati rendszereknek. Lehetővé teszi, hogy a rendszereket „szűkre szabva" a szolgáltatásoknak csak a működésükhöz feltétlenül szükséges jogosultságokat adjuk meg. A lehető legkevesebb jogosultság alapelvének ilyen jellegű megvalósításával meg lehet előzni a biztonsági határok hibás vagy rosszindulatú kód, vagy hibát vétő vagy rosszakaratú felhasználó miatt előforduló átlépését.

Egy külső személyek számára szolgáltatásokat nyújtó webkiszolgálót például számos módon lehet védeni:

- Szükségtelen szolgáltatások letiltása
- A kiszolgáló program ***chroot börtönben (jail)*** való futtatása
- Helyi csomagszűrés ***iptables*** segítségével
- Jogosultságkezelés ***sudo*** segítségével
- Beállító fájlok zárolása

A fentiekkel több réteggel érjük el a biztonság növelését, tulajdonképpen a mélybe nyúló védelem alapelvét követjük. Az ***SELinux*** a rendszert egy további biztonsági réteggel bővíti, a ***kötelező hozzáférés-vezérléssel (mandatory access control, MAC)***. A normál ***SELinux*** a ***MAC***-et ***típusszabályokkal (type enforcement, TE)*** és ***szerep alapú hozzáférés-vezérléssel (role-based access control, RBAC)*** valósítja meg, mindezt egy központi kezelésű biztonsági házirend vezérli, melynek betartatásáról a rendszermag gondoskodik. A hagyományos ***UNIX*** biztonságtól eltérően a normál felhasználók semmilyen ellenőrzési jogot nem kapnak az ***SELinux*** biztonsági házirendjére (erre utal a kötelező jelző), míg a rendszergazda, a root feladatköre több, egymástól elkülönülő felügyeleti szerepre is feloszthatő, amit a feladatok és a felelősség szétosztásának nevezünk.

A hagyományos, ***önkényes hozzáférés-vezérlést (discretionary access control, DAC)*** a ***TE*** modell megszigorítja, ugyanis az operációs rendszer egyes objektumaihoz – folyamatokhoz, fájlokhoz és hálózati erőforrásokhoz – típusokat rendel, majd pontosan körülírja a közöttük folyó párbeszédekre vonatkozó szabályokat. (Egy folyamat típusát általában tartománynak nevezik.) Mindezekkel az eszközökkel kifinomult hozzáférés-vezérlés valósítható meg, és a lehető legkevesebb jogosultság elve az operációs rendszer általános értelemben vett megerősítésén túl is messze kiterjeszthető.

Az SELinux hálózati hozzáférés-vezérlési alrendszere

Az ***SELinux*** a 2.6-os rendszermag ***LSM (Linux Security Modules, Linux biztonsági modulok)*** és ***Netfilter API***-jaira épül. Az ***LSM*** és a Netfilter egyaránt egy a rendszermag stratégiai pontjain elhelyezett csatlakozási pontokból, „kampókból" álló hozzáférés-vezérlési keretrendszer. A rendszermagon belüli adatáramlások a csatlakozási pontokon keresztül a biztonsági modulok, például az ***SELinux*** felé irányulnak, ezek elvégzik a hozzáférés-vezérlési számításokat, majd közlik az eredményt a csatlakozási ponttal. A csatlakozási pont a biztonsági modul utasítása alapján vagy engedélyezi az adatáramlást, vagy megtiltja azt.

Az ***SELinux*** egyik fontos tervezési alapelve, hogy az operációs rendszer objektumainak szintjén ellenőrzi a hozzáférést. Ahelyett, hogy egy biztonsági figyelő egyszerűen csak megmondaná, adott program adott kapcsolókkal és átadott értékekkel elindíthat-e egy rendszerhívást vagy sem, az ***SELinux*** a program futása során annak teljes biztonsági környezetét átlátja, továbbá hozzáfér az elérni kívánt objektum és a végrehajtani kívánt művelet biztonsági címkéjéhez is. A rendszergazda által futtatott `ls` parancs például teljesen más, mint a normál felhasználók által használt.

Egy ***SELinux*** engedély általános formátuma a következő:

- `művelet (forrás környezet)`
- `(cél környezet):(cél objektumosztályok)`
- `engedélyek`

Egy példa az SELinux házirendjéből:

```
allow bluetooth_t self:socket listen;
```

A fentiek értelmében a `bluetooth_t` tartomány hallgatózási (listen) engedélyt kap a saját biztonsági környezetével (kontextusával) felcímkézett foglalatokhoz (socket). Egy a `bluetooth_t` tartományban futó folyamat tehát egy általa

birtokolt foglalaton jogosult a listen() hívás alkalmazására. A `self` (önmaga) csak egy egyszerű jelölés a cél környezet egyenlővé tételére a forrás környezettel. A foglalatokkal kapcsolatos házirendben gyakran láthatunk ilyen parancsokat, ugyanis a foglalatok általában a létrehozó folyamattal azonos biztonsági környezet címkét kapnak.

A hálózati objektumok címkézése

SELinux alatt az objektumok biztonsági címkéje a következő formátumot követi:

```
felhasználó:szerep:típus
```

Például:

```
root:staff_r:staff_t
```

Ez egy olyan folyamat biztonsági környezete, amelyet a `staff_r` szerepen keresztül, a `staff_t` tartományban a `root` futtat. A 80-as kapuhoz tartozó címke a következő:

```
system_u:object_r:http_port_t
```

A `system_u` felhasználó és az `object_r` szerep a rendszerobjektumok esetében alapértelmezett. Semmi értelme nem volna egy kapuhoz valódi felhasználónevet vagy szerepet rendelni, hiszen senki sem birtokolja, és nem kezdeményez az ***SELinux*** elbírálását igénylő műveleteket sem.

Foglalatok

A foglalatok címkézése a hozzájuk tartozó fájlleíró (i-node) alapján történik. Egy foglalat lehet általános, vagy tartozhat az alábbi foglalat alosztályok valamelyikébe:

- ***UNIX*** folyam (***UNIX*** stream)
- ***UNIX*** datagram
- ***TCP***
- ***UDP***
- Nyers (***ICMP*** vagy egyéb nem ***TCP***/***UDP***) (***Raw***)
- ***Netlink*** család
- Csomag (Packet)
- Kulcs (***pfkeyv2***) (***Key***)

A biztonsági házirendben a foglalatok alosztályait is meg lehet különböztetni egymástól, így magas fokú rugalmasságot lehet elérni, és kifinomult vezérlést lehet megvalósítani a különféle hálózati protokollok felett:

```
allow lpd_t printer_port_t:tcp_socket name_bind;
```

A szabály értelmében csak az `lpd_t` tartományban létrehozott ***TCP*** foglalatok kötődhetnek a `printer_port_t` típusú kapukhoz.

Kapuk

Az ***IPv4*** és az ***IPv6*** kapuk címkézése implicit módon, a rendszermagon belül, a házirend által megszabottak szerint történik. A kaputípusok címkézésének formátuma a következő:

```
portcon protokoll { kapuszám | kaputartomány }
        környezet
```

Az alábbi szabály a szabványos nyomtatókapu címkézéséhez adja meg a biztonsági környezetet:

```
portcon tcp 515 system_u:object_r:printer_port_t
```

Hálózati csatolók

Minden hálózati csatoló (network interface, netif) a házirendben megadottak szerint kap biztonsági környezetcímkét. A hálózati csatolók címkézése a következőképpen történik:

```
netifcon csatoló környezet
alapértelmezett_üzenet_környezet
```

Az `alapértelmezett_üzenet_környezet` érték a csatolón keresztül beérkező üzenetek címkézésére szolgálna, ám jelenleg használaton kívül van.
Két példa házirend netif címkéjére:

```
netifcon eth0 system_u:object_r:netif_intranet_t
➥[...]
netifcon eth1 system_u:object_r:netif_extranet_t
➥[...]
```

Csomópontok

SELinux alatt a csomópont egy ***IPv4*** vagy ***IPv6*** címet vagy hálózati maszkot jelent. Lehetővé teszi állomás- és hálózatcímek házirenden keresztül való biztonsági címkézését. A csomópontok címkézése a következő formátumot követi:

```
nodecon cím maszk környezet
```

Példa helyi cím címkézésére:

```
nodecon 127.0.0.1 255.255.255.255
        system_u:object_r:node_lo_t
nodecon ::1
➥ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
        system_u:object_r:node_lo_t
```

Hálózati csatlakozási pontok és engedélyek

A hozzáférés-vezérlési csatlakozási pontokat minden foglalatokkal kapcsolatos rendszerhíváshoz megvalósították, így minden foglalat alapú hálózati protokoll működését ellenőrizni lehet ***SELinux*** házirenddel. A csatlakozási pontok némelyike csupán a rendtartást szolgálja, ám nagyobb részük egy vagy több hozzáférés-vezérlési engedély ellenőrzését végzi. Általános foglalatvezérlő meglehetősen sok van, ezért a csatlakozási pontok, a foglalatokra vonatkozó rendszerhívások és az engedélyek közötti kapcsolatokat az *1. táblázatban* foglaltam össze.
Belül a `socket()` rendszerhívás két csatlakozási pontra oszlik. Az `selinux_socket_create` csatlakozási pont annak ellenőrzésére alkalmas, hogy a folyamat létrehozhatja-e a megadott típusú foglalatot. Az `selinux_socket_post_create` felügyeleti csatlakozási pont a foglalathoz rendelt fájlleíró biztonsági címkéjének megadását és foglalattípusának meghatározását teszi lehetővé.
Az ***SELinux*** engedélyek a rendszerhívások és az egyéb műveletek biztonsági szemszögből való elvonatkoztatásához is hozzájárulnak. Vegyük észre például, hogy

1. táblázat *A csatlakozási pontok, a foglalatokra vonatkozó rendszerhívások és az engedélyek kapcsolatai*

Csatlakozási pont	Rendszerhívás	SELinux engedély
`selinux_socket_create`	`socket`	`create` (létrehozás)
`selinux_socket_post_create`	`socket`	n. a.
`selinux_socket_bind`	`bind`	`bind` (kötődés)
`selinux_socket_connect`	`connect`	`connect` (kapcsolódás)
`selinux_socket_listen`	`listen`	`listen` (hallgatózás)
`selinux_socket_accept`	`accept`	`accept` (fogadás)
`selinux_socket_sendmsg`	`sendmsg, send, sendto`	`write` (írás)
`selinux_socket_recvmsg`	`recvmsg, recv, recvfrom`	`read` (olvasás)
`selinux_socket_getsockname`	`getsockname`	`getattr` (jellemzők lekérdezése)
`selinux_socket_getpeername`	`getpeername`	`getattr` (jellemzők lekérdezése)
`selinux_socket_setsockopt`	`setsockopt`	`setopt` (kapcsolók megadása)
`selinux_socket_getsockopt`	`getsockopt`	`getopt` (beállítások lekérdezése)
`selinux_socket_shutdown`	`shutdown`	`shutdown` (leállítás)

2. táblázat *A fájlokra jellemző csatlakozási pontok és engedélyek*

Csatlakozási pont	Rendszerhívás	SELinux engedély
`selinux_file_ioctl`	`ioctl`	`ioctl`
`selinux_inode_getattr`	`fstat`	`getattr` (jellemzők lekérdezése)
`selinux_inode_setattr`	`fchmod, fchown`	`setattr` (jellemzők beállítása)
`selinux_file_fcntl`	`fcntl`	`lock` (zárolás)
`selinux_file_lock`	`fcntl, flock`	`lock` (zárolás)
`selinux_file_permission`	`write, write, read`	`append, write, read` (hozzáfűzés, írás, olvasás)

a `getsockname()` és a `getpeername()` rendszerhíváshoz egyaránt a `getattr` engedély tartozik. Az ***SELinux*** ezek biztonsági szerepét egyenlőnek veszi. Hasonlóan, minden `sendmsg()` és `recvmsg()` alapú rendszerhívás biztonságfelügyeleti szempontból egyszerű írásnak vagy olvasásnak számít. Akit a részletek is érdekelnek, a csatlakozási pontokat megvalósító kódot a 2.6-os rendszermag forrásában, a ***security/selinux/hooks.c*** fájlban találja.

Mivel a foglalatok is fájlok, a fájlokra vonatkozó hozzáférésvezérlés egy részét is megöröklik. A 2. *táblázat* a fájlokra jellemző, de a foglalatokra is érvényes csatlakozási pontokat tartalmazza.

A UNIX tartományra vonatkozó engedélyek

Linux alatt a ***UNIX*** tartomány foglalatait elvonatkoztatott névtérben, a fájlrendszertől függetlenül lehet létrehozni. Az elvonatkoztatott névtérben lévő ***UNIX*** tartománybéli foglalatok közötti adatcserék, illetve ezek irányultságának ellenőrzésére további csatlakozási pontokat valósítottak meg. Az `selinux_socket_unix_stream_connect` csatlakozási pont a `connectto` engedély meglétét ellenőrzi, amikor egy ***UNIX*** tartománybéli foglalat adatfolyam (***stream***) kapcsolatot próbál létesíteni egy másikkal.

Az `selinux_socket_unix_may_send` csatlakozási pont a `sendto` engedély meglétét követeli meg ahhoz, hogy az egyik ***UNIX*** tartománybéli foglalat datagramot küldhessen egy másiknak.

A ***UNIX*** tartománybéli foglalatok egy további ***Linux*** alatti szolgáltatása az egyenrangú felek hitelesítése a `SO_PEERCRED` – foglalatokra vonatkozó – kapcsolóval. A kapcsoló az egyenrangú fél felhasználó-, csoport és folyamatazonosítójának lekérdezését eredményezi. ***SELinux*** alatt egy-egy egyenrangú fél biztonsági környezetét egy új, szintén foglalatokra vonatkozó művelettel, a `SO_PEERSEC` kapcsolóval is megtudhatjuk. A `getsockopt(2)` hívást ezzel a kapcsolóval indítva a `selinux_socket_getpeersec` csatlakozási pontra adhatjuk át a vezérlést, mely a felhasználó által megadott pufferbe másolja a biztonsági környezetet. Ezt a megoldást helyi folyamatok közötti kommunikációra, például ***megnövelt biztonságú adatbusz (Security-Enhanced DBUS)*** megvalósítására használják.

Netlink vezérlők

A ***Netlink*** foglalatok üzenet alapú felhasználó/rendszermag adatcserét tesznek lehetővé. Például a rendszermag irányítótáblájának és az ***IPSec*** működésének beállítására használhatók.

A *Netlink* adatcsere aszinkron jellegű, az üzenetek küldése és fogadása eltérő biztonsági környezetben is lehetséges. *Netlink* csomag továbbításakor a küldő biztonsági engedélyei egy képességhalmaz formájában a csomag mellé kerülnek, fogadáskor pedig megtörténik az ellenőrzésük. Így lehetővé válik, hogy például a rendszermag forgalomirányító kódja ellenőrizze, hogy az irányítótábla frissítését küldő felhasználónak valóban van-e joga ilyen frissítés küldésére.
Az *LSM* tervezet részeként a képességeket kezelő kódok a rendszermagból átkerültek egy biztonsági modulba, így az *LSM*-ek szükség esetén többféle biztonsági modell megvalósítására is alkalmasak.
Az *SELinux* modul az `selinux_netlink_send` csatlakozási pontot használja arra, hogy kizárólag a `NET_ADMIN` képességet a rendszermag által küldött *Netlink* csomagokba bemásolja. Az `selinux_netlink_recv` csatlakozási pont a biztonsági szempontból fontos üzenetek fogadásakor jut szerephez. Az *SELinux* ezt a csatlakozási pontot alkalmazza annak ellenőrzésére, hogy a `NET_ADMIN` képesség másolása a csomag elküldésekor megtörtént-e, valamint a küldő folyamat valóban rendelkezett-e ezzel a képességgel.
Egyre több *Netlink* család kerül megvalósításra, és az *SELinux* a biztonsági szempontból kritikus *Netlink* foglalatokhoz alosztályokat ad meg. Így a foglalatok feletti felügyelet az egyes *Netlink* családokra egyedileg állítható be, amivel elérhető például az irányítási üzenetek és a rendszer naplózási üzeneteinek megkülönböztetése.
Az *SELinux* az `selinux_netlink_send` csatlakozási pont révén annak meghatározására is képes, hogy a *Netlink* foglalatok egyes típusain az üzenetek olvasási vagy írási műveletek, majd rendre alkalmazza az `nlmsg_read` vagy az `nlmsg_write` engedélyeket. Ezáltal rendkívül kifinomult házirend állítható össze, például adott tartomány számára engedélyezni lehet az irányítótábla kiolvasását, miközben meg lehet tiltani annak frissítését.

IPv4 és IPv6 vezérlők

Az *SELinux* számos vezérlőt tartalmaz a *TCP*, az *UDP* és a *nyers (raw)* foglalat alosztályokhoz. A `node_bind` engedély szabja meg, hogy egy foglalat kötődhet-e egy megadott típusú csomóponthoz. Értelemszerűen csak helyi IP-címnél érdemes használni, és általa adott démon adott IP-címhez való kötődését lehet meggátolni.
A `name_bind` engedély azt szabja meg, hogy egy foglalat kötődhet-e adott típusú kapuhoz. Csak akkor jut szerephez, ha a kapu száma a helyi kaputartományon kívülre esik.
A helyi kaputartomány az, ahonnan a rendszermag a kapuszámok önműködő kiosztását végzi (például, amikor egy kimenő *TCP*-kapcsolat forráskapujának számát kiválasztja), megadására a `net.ipv4.ip_local_port_range` rendszerhívás használható. Egy átlagos rendszeren a tartomány a következő:

```
$ sysctl net.ipv4.ip_local_port_range
net.ipv4.ip_local_port_range = 32768    61000
```

A `name_bind` meghívására tehát csak akkor kerül sor, ha egy foglalat egy e tartományon kívülre eső kapuhoz kötődik. Az *SELinux* az 1024 alatti számú kapuk esetében mindig megvizsgálja az engedélyeket, függetlenül a sysctl beállítástól. Mindkét kötődés vonatkozású vezérlő az `selinux_socket_bind` csatlakozási pontról kerül meghívásra, mely viszont a ***bind(2)*** rendszerhíváson keresztül jut a vezérléshez.
A `send_msg` és a `recv_msg` engedély annak szabályozására alkalmas, hogy egy foglalat egy adott típuson vagy kapun keresztül küldhet vagy fogadhat-e üzeneteket.
Számos vezérlő határozza meg, hogy *TCP*, *UDP* vagy nyers foglalaton keresztül lehet-e megadott típusú netif és csomópont objektumok felé és felől csomagokat küldeni és fogadni; ezek a `tcp_send`, a `tcp_recv`, az `udp_send`, az `udp_recv`, a `rawip_send` és a `rawip_recv`.
Mindezeket az üzenet alapú vezérlőket az `selinux_sock_rcv_skb` csatlakozási pont hívja meg a beérkező üzenetekre, ez ugyanis a hálózati verem első olyan pontja, ahol a csomagok egyértelműen hozzárendelhetők a fogadó foglalatokhoz. A kimenő csomagok kezeléséhez az *SELinux* bejegyez egy *Netfilter* csatlakozási pontot, és az IP rétegben elfogja a csomagokat. A kimenő csomagokhoz csatolt tulajdonlási adatok ebben a fázisban is megmaradnak.
A fenti vezérlők annyiban mind protokollfüggetlenek, hogy *IPv4* és *IPv6* protokollokkal egyaránt működnek.

Hálózati házirend

Az elmélettel foglalkoztunk eleget, lássunk tehát egy valódi *SELinux* házirendet egy egyszerűbb hálózati alkalmazáshoz. Mivel helyszűkében vagyunk, az igazi hálózatkezelés pedig sosem egyszerű, csak egy egyszerű *TCP* visszhang-ügyfél számára fogunk házirendet készíteni.
Az ügyfél forráskódja a cikkhez tartozó internetes források között szereplő weboldalon érhető el. Rövid annyit róla, hogy létrehoz egy *TCP* foglalatot, kapcsolódik a távoli állomás visszhang kapujához, kiír valami szöveget, majd visszaolvassa.
Saját munkaállomásom két *Ethernet* csatolóval rendelkezik, a példában az `eth0` egy intranetbe tartozik, és a kiszolgáló, melyhez csatlakozok, a ***10.3.1.2*** IP-címet viseli.
Amit a biztonsági házirendtől elvárunk:

- Az ügyfél az operációs rendszernek csak a valóban szükséges erőforrásaihoz férjen hozzá.
- Az ügyfél kommunikációs lehetőségei csak a ***10.3.1.0/24*** alhálózatra eső `inetd` kiszolgálókra és az `eth0` csatolóra terjedjenek ki.

Házirend

Az alábbiakban egy a fentieknek megfelelő, megjegyzésekkel ellátott házirend szerepel. Használatához telepíteni kell a terjesztésünkhöz készült *SELinux* házirend forráscsomagokat, majd legfelső szintű könyvtárába kell lépnünk (saját gépemen ez a */etc/selinux/strict/src/policy*).
Hozzuk létre a domains/program/echoclient.te nevű fájlt, majd adjuk hozzá az *1. kódrészletben* szereplő házirendbejegyzéseket.
A *net_contexts* fájlhoz a következő címkemegadásokat kell hozzáadni:

```
# eth0 címke
netifcon eth0 system_u:object_r:netif_intranet_t
          system_u:object_r:unlabeled_t
  # A belső hálózat címkézése.
```

1. kódrészlet echoclient.te

```
# Egyszerű visszhangügyfél (echoclient) házirend
# a linuxvilágos cikkhez
# Fájl: domains/program/echoclient.te

# Az echoclient_t típus megadása tartományként
type echoclient_t, domain;

# Az echoclient_exec_t megadása futtatható típusú
# fájlként.
type echoclient_exec_t, file_type, exec_type;

# Ez egy makró, ez fogja engedélyezni
# a megfelelően címkézett futtatható fájlnak
# az átlépést az echoclient_t tartományba
# a staff_t tartományból.
domain_auto_trans(staff_t, echoclient_exec_t,
                  echoclient_t)

# Megadjuk, hogy mely szerepek léphetnek be az
# echoclient_t
# tartományba.
role staff_r types echoclient_t;

# Ez a makró teszi lehetővé a tartománynak
# a megosztott
# könyvtárak használatát.
uses_shlib(echoclient_t);

# Azoknak az engedélyeknek a biztosítása, amelyek
# a program futtatásához staff_t-ként, SSH-n
# keresztül való bejelentkezésnél szükségesek,
# így válik lehetővé a hibakereső és -jelző
# üzenetek kiírása a felhasználó termináljára.
allow echoclient_t sshd_t:fd use;
allow echoclient_t staff_devpts_t:chr_file {
                                getattr read write };

# Hálózati beállítások
# Ezek a tartomány által igényelt foglalat
# engedélyek.
# Érdemes megjegyezni, hogy csak a TCP
# foglalatokra érvényesek.
allow echoclient_t echoclient_t:tcp_socket {
            connect create read shutdown
            ➥write };
# Engedélyezzük a programnak, hogy TCP üzeneteket
# küldjön a visszhangkapu
# felé, illetve ilyeneket fogadjon tőle.
# Egy normál házirendben a kapu
# inetd_port_t címkét kap, és az inetd által
# kezelt kapuk csoportjába tartozik. A
# net_contexts fájlban szereplő házirendet
# módosítva a szabály
# egyetlen kapura is szűkíthető.
allow echoclient_t inetd_port_t:tcp_socket {
                                recv_msg send_msg };
# Az intranetes csatolón keresztül csak a TCP
#forgalmat engedélyezzük.
allow echoclient_t netif_intranet_t:netif {
                                tcp_recv tcp_send };

# A belső IP-címekkel csak TCP alapú
# kommunikációt
# engedélyezünk.
allow echoclient_t node_internal_t:node {
                                 tcp_recvtcp_send };
```

```
nodecon 10.3.1.0 255.255.255.0
        system_u:object_r:node_internal_t
```

A ***types/network.te*** fájlba kerülő sorok:

```
# A netif_intranet_t megadása hálózati
# csatoló típusként.
type netif_intranet_t, netif_type;
```

Fájlkörnyezet megadása a futtatható fájl számára egy új, ***file_contexts/program/echoclient.fc*** nevű fájlban:

```
# Alapértelmezett fájlkörnyezet a címkézéshez
/tmp/echoclient –
➥system_u:object_r:echoclient_exec_t
```

Fordítsuk le és töltsük be a házirendet:

```
$ make load
```

Ezzel a házirend elkészült. Látszólag rengeteg munka volt vele, ám ha már otthonosan mozgunk a különféle házirend-fájlok között, és jobban megismertük a szükséges házirend-bejegyzések típusait, minden könnyebbé válik. Ekkor már az olyan eszközöket is könnyebben tudjuk majd használni, mint például az `audit2allow`, amely a napló tiltási üzenetei alapján engedélyező szabályokat hoz létre. A mindennapi házirendkészítéshez jobb valamilyen magasabb szintű, grafikus eszközt használni, jelen esetben azonban a dolgok működésének lépésről lépésre való bemutatása volt a cél.

Próba

Fordítsuk le és lássuk el címkével a futtatható ügyfél-programot:

```
$ make echoclient
cc      echoclient.c    -o echoclient

$ restorecon /tmp/echoclient
```

Ellenőrizzük a címke helyességét:

```
$ getfilecon /tmp/echoclient
/tmp/echoclient
➥system_u:object_r:echoclient_exec_t
```

Erre a célra az `ls -Z` parancsot is használhatjuk.
Lássuk, működik-e a rendszer. Rootként jelentkezzünk be SSH-n keresztül staff_r szerepbe, majd:

```
$ id -Z
root:staff_r:staff_t
$ /tmp/echoclient 10.3.1.2
Sending message: 'Hello, cliche'
Received message: 'Hello, cliche'
```

Működik!
A házirendet auditallow szabályokkal bővítve azt is figyelemmel követhetjük, hogyan folyik az engedélyek megadása.
Ellenőrizzünk is néhány házirendbéli szabályt, vajon valóban működik-e.

1) Próbáljunk egy az intraneten kívülre eső IP-címmel kapcsolatot létesíteni. A címet helyileg irányítsuk, így nem fordulhat elő, hogy véletlenül az internet felé küldünk el egy csomagot:

```
$ ip ro add 196.40.74.92 via 10.3.1.2 dev eth0
$ /tmp/echoclient 196.40.74.92
```

A program *TCP* időtúllépést kap, és csomag küldésekor az alábbi, tiltásról értesítő naplóüzenet jön létre:

```
avc:  denied  { tcp_send } for  pid=10831
  exe=/tmp/echoclient saddr=10.3.1.1 src=32822
  daddr=196.40.74.92 dest=7 netif=eth0
  scontext=root:staff_r:echoclient_t
  tcontext=system_u:object_r:node_t
  tclass=node
```

Mint vártuk, az echoclient_t tartomány nem kapott engedélyt arra, hogy *TCP* csomagot továbbítson */node_t/* csomópontnak (ez az alapértelmezett általános csomópont környezet).

2) Próbáljunk másik csatolón keresztül kommunikálni. A visszhang kiszolgáló IP-címét irányítsuk a hurok felületen keresztül, így a csomagok erre kerülnek elküldésre:

```
  $ ip ro add 10.3.1.2 via 127.0.0.2 dev lo

  $ /tmp/echoclient 10.3.1.2
avc:  denied  { tcp_send } for pid=10828
  exe=/tmp/echoclient saddr=10.3.1.1 src=32821
  daddr=10.3.1.2 dest=7 netif=lo
  scontext=root:staff_r:echoclient_t
  tcontext=system_u:object_r:netif_lo_t
  tclass=netif
```

Most is helyes eredményt kaptunk. Az `echoclient_t` tartomány nem kapott engedélyt arra, hogy `netif_lo_t` hálózati csatolón keresztül továbbítson csomagot.
Az `echoclient` program a házirend értelmében rendkívül szűkre szabott jogosultságokkal fut. Amit nem engedtünk meg kifejezetten, az tilos számára. A programban esetlegesen felmerülő hibák, a felhasználó hibázásának vagy rosszindulatának hatása a házirend révén erőteljesen korlátozható.
Egyszerű példánk alapján világossá válik, hogy ***SELinux*** házirenddel hogyan lehet elérni a hálózati biztonság terén kitűzött célokat. Egy valós környezetben alkalmazott házirendnek számos különleges lehetőséget kell biztosítania, ezekkel a helyszűke és az érthetőség miatt nem foglalkoztunk, de példaként érdemes az ***ICMP***-üzenetek és a ***DNS***-lekérdezések engedélyezését megemlíteni. Mindenkinek javaslom a saját terjesztéséhez készült házirendforrások tanulmányozását, illetve a grafikus házirendkészítő alkalmazások kipróbálását.

Jövőbeli fejlesztések

SELinux alatt valószínűleg meg fog valósulni a címkézett hálózatkezelés valamilyen formája. Ennél a megoldásnál maga a hálózati forgalom kerül címkézésre – ilyesmire leginkább bizalmas adatokat kezelő katonai és kormányzati rendszerekben láthatunk példát. Az ***SELinux*** egy korábbi változata IP-beállításokat használt a csomagok címkézésére, ám ezt a megoldást kivették belőle, még mielőtt a rendszermag fő ágába bekerült volna, ugyanis a szükséges csatlakozási pontok túlságosan mélyrehatók voltak. Egy másik megoldás lenne az ***SELinux*** és az ***IPSec*** egybeépítése, és a ***biztonsági társulások (Security Associations, SA)*** címkézése a csomagok helyett. Egy adott ***SA***-n érkező csomagot implicit módon lehetne címkézni az ***SA*** környezetével. A korábbi ***Flask Project*** keretein belül már készült minderre egy prototípus, az itt szerzett eredmények kiváló útmutatóként szolgálhatnak.
Az ***SELinux*** szorosabb egybeépítése más hálózatbiztonsági összetevőkkel, például a titkosítással vagy a tűzfalakkal, további kutatások tárgya.

Köszönetnyilvánítás

Köszönettel tartozom ***Russell Cokernek*** írásom átnézéséért és értékes megjegyzéseiért.

Linux Journal 2005. január, 129. szám

James Morris (jmorris@redhat.com) az ausztráliai Sydney városából származó rendszermag-programozó, jelenleg Bostonban, a Red Hatnél dolgozik. Az SELinux, a Networking és a Crypto API rendszermag karbantartója; LSM fejlesztő és az Emeritus Netfilter Core Team tagja.

KAPCSOLÓDÓ CÍMEK

- www.nsa.gov/selinux
- www.lurking-grue.org/gettingstarted_newselinuxHOWTO.html
- people.redhat.com/kwade/fedora-docs/selinux-faq-en
- www.gentoo.org/proj/en/hardened/selinux