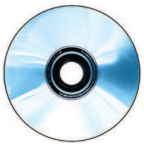
Az osztott linuxos biztonsági modell



Hozzáférés-ellenőrzés megvalósítása Linux-telepekre kihegyezett rendszermag-bővítmény segítségével.

A távközlési ipar magas igényeinek kielégítésére hagyományosan nagy megbízhatóságú, magas rendelkezésreállású és méretezhető telepeket használ oly módon, hogy közben költséghatékony gép- és programmegoldásokat alkalmaz. Ezért gondolni kell a telepek biztonságának megteremtésére is, azonban erre eleddig még nem született hatékony megoldás.

A távközlési iparból hiányzó biztonságos telepmegoldások pótlására az Ericsson Researchöz tartozó Open Systems Lab (Montreal, Kanada) egy új projektet indított útjára, amelyet osztott biztonsági háttérnek nevez (Distributed Security Infrastructure – DSI). A DSI elsődleges céljának tekinti az olyan fejlett biztonsági hátterek tervezését és megvalósítását, amelyekkel a távközlési iparban használt linuxos telepek biztonsága megteremthető (lásd bővebben a 24. oldalon). A DSI egyik fontos eleme az osztott biztonsági modul (Distributed Security Module – DSM), ami a linuxos telepeken a kötelező érvényű jogosultságellenőrzést valósítja meg.

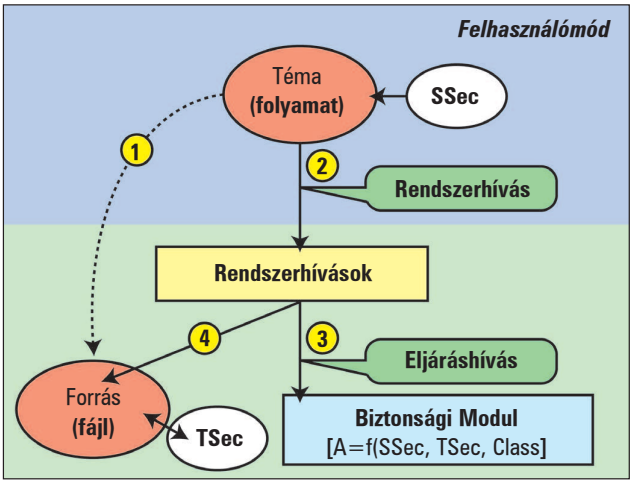
Ebben a cikkben azt tárgyaljuk, hogy milyen céllal alkalmazhatjuk az osztott biztonsági bővítményt, milyen lehetőségei vannak, továbbá tulajdonságait, felépítését és megvalósításának jelenlegi állapotát is bemutatjuk. Egyúttal ismertetjük azt is, hogyan telepíthetünk DSM-et, és miként próbálhatjuk ki a működését.

Kötelező érvényű jogosultság-ellenőrzés

A jelenleg alkalmazott biztonsági megoldások valamilyen különálló jogosultság-ellenőrzésen alapulnak. Ezek a módszerek azonban alkalmatlanok a bonyolult rendszerek védelmére, arra, hogy a napjainkban használt fejlett támadási típusok ellen védekezzünk. A jogosultságok megítélése a felhasználó azonosításán és hovatartozásán alapszik. Ennek következtében ez az ellenőrzés könnyűszerrel megkerülhető, és az alantas szándékkal írt programok egyszerűen okozhatnak leállásokat és üzemzavarokat a rendszerben. A különböző kutatások kimutatták, hogy az operációs rendszer által biztosított kötelező érvényű jogosultság-ellenőrzés a teljes rendszer szempontjából nézve nélkülözhetetlen. Egyúttal az is kiderült, hogy egy ilyen jogosultság-ellenőrzési módszerrel a számítógépes környezetben pontosan szabályozhatók az egységek közötti bonyolult kapcsolatok.

A DSI projekt részeként megvizsgáljuk a jogosultsági rendszerhez létrehozott keretrendszer szerkezetét és felépítését. Létrehozunk egy olyan Linux-rendszermagmodult, ami telepeken valósít meg jogosultság-ellenőrzést. Az ezen a területen végzett munkánkkal azt szeretnénk elérni, hogy Linux-telepek esetében is megbízható operációs rendszer legyen.

Munkánk elsősorban a Flask-szerkezetre és a Linux biztonsági modul (LSM) keretrendszerre épül. Mi azonban nem a különálló rendszerekre összpontosítunk, hanem a telepekre épülő megoldásokra. Megvizsgáljuk a biztonsággal együtt járó teljesítménygondokat is, mivel egy-egy biztonsági megoldás olykor jelentős teljesítménybeli visszaeséssel járhat, ugyanakkor a felhasználó számára egyéb zavaró következményei is lehetnek.



1. ábra Jogosultság-ellenőrzés az LSM-bővítménnyel

DSM-megvalósításunk egyik legfontosabb tulajdonsága, hogy osztott modellen alapul. Emiatt a biztonsággal kapcsolatos egységek elhelyezkedése a rendszerben a telep biztonságának szempontjából nézve nem lényeges.

A Linux biztonsági modul (LSM)

Az LSM-keretrendszer nem biztosít semmiféle kiegészítő védelmet a rendszermagban, ehelyett a biztonságos modulok fejlesztéséhez szükséges hátteret adja. Az LSM rendszermagfolt biztonsági területeket ad hozzá a rendszermag adatterületeihez és rendszerhívásaihoz (vagyis az úgynevezett kapcsokhoz), így biztosítva a modulszintű hozzáférés-védelmet.

Az LSM tagfüggvényeket biztosít biztonsági modulok hozzáadására és elvételére, valamint egy általános biztonsági rendszerhívás is létezik, ami a felhasználói programok és az LSM-re épülő biztonsági bővítmények közti kapcsolattartásért felelős.

Minden egyes LSM-kapocs (hook) egy függvénymutatóval rendelkezik egy teljes hatókörű, `security_ops` nevű szerkezetben. Mivel ezek a függvények be vannak ágyazva a rendszermagba, és még egy biztonsági kiterjesztés beillesztése előtt is meghívódnak, kezdetben a függvények elérését tároló szerkezet az úgynevezett helyfoglaló függvényekre mutat.

Ha betöltünk egy biztonsági bővítményt, az ezeket a helyfoglaló függvényeket a saját függvényeire cseréli le, amik már a tényleges biztonságot szolgálják. A helyfoglaló függvényeket a `register_security` tagfüggvénnyel lehet lecserélni a modul saját függvényeire.

Az `unregister_security` tagfüggvény ennek az ellenkezőjét teszi, vagyis a hivatkozásokat az egyszerű helyfoglaló függvényekre mutatókra cseréli le.

Az LSM-tagfüggvények két csoportra oszthatók:

1. a biztonsági területek kezeléséért felelős kapcsokra és
2. a jogosultságok kezeléséért felelős kapcsokra.

Minden egyes létrejövő Linux-erőforráshoz egy biztonsági címke kapcsolódik. Ezeknek a címkéknek a segítségével ellenőrizhetők a biztonsági kapcsokhoz fűződő jogosultságok. Ha az erőforrás megsemmisül, vele együtt semmisül meg a hozzátartozó címke is. A biztonsági területek kezeléséért felelős kapcsok felelősek a címkék létrehozásáért és eltávolításáért is. Erre a `task_security_ops` szerkezetben találhatunk példát, ilyen az `alloc_security` és a `free_security` függvény. Az LSM-féle jogosultságkiosztás folyamata az *1. ábrán* látható. Tegyük fel, hogy egy igénylő (jelen esetben egy folyamat) egy `SSec` biztonsági azonosítóval rendelkezik, és megpróbál hozzáférni (`access(1)`) egy `TSec` biztonsági azonosítóval rendelkező erőforráshoz (esetünkben egy fájlhoz). A rendszerhívást a Linux-rendszermag fogja kezelni (a rendszerhívási felület az 1. ábrán látható). Mielőtt a rendszermag döntene a művelet engedélyezéséről, kapcsolatba lép (a kapcsokon keresztül) az LSM-modullal (3.), ahol a felhasználóhoz kapcsolódó jogok kiosztása egy `f` függvényen keresztül valósul meg. Az LSM kiértékeli az `f` függvényt, az eredményt pedig visszaküldi a rendszermagnak. Ezután a rendszermag vagy engedélyezi a hozzáférést az erőforráshoz, vagy nem (4.).

Az osztott biztonsági modul (DSM)

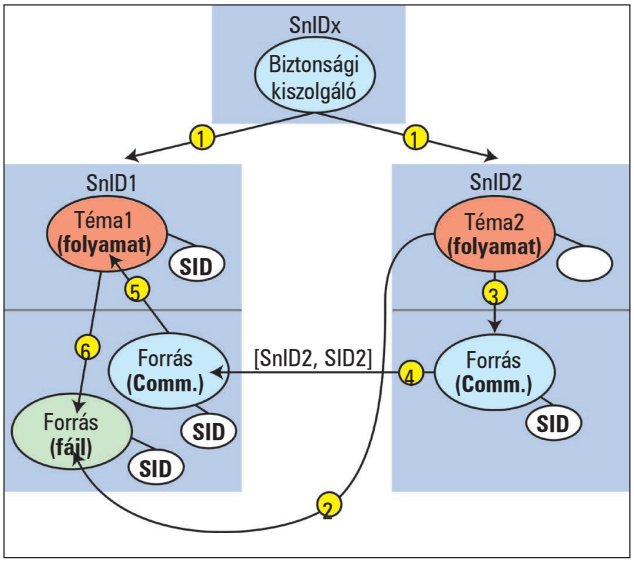
A DSM a DSI részét képezi. A DSM megvalósításának célja a hozzáférés-szabályozás kikényszerítése, és a küldő gép és folyamat biztonsági azonosítóival a telep csomópontjai között az IP-üzenetek címkézése. A DSM fejlesztését a 2.4.17-es Linux-rendszermaggal és az ahhoz tartozó magfolttal (`lsm-full-2002_01_15-2.4.17.patch`) kezdtük. A DSM megvalósítása a CIPSO-n és a FIPS-188 szabványokon alapult, amely az IP fejlécének módosítását (vagyis az IP-fejléchez való kapcsolók hozzáadását), és a kapcsok hozzáadását szabályozza.

Osztott biztonsági háttér (DSI)

Mivel a DSM a DSI része, erről már az előző cikkünkben is olvashattak. Egy adatátvitelre kiélezett Linux-telep részeként a DSI-nek átvitelalapú követelményeknek kell megfelelnie: nevezetesen a megbízhatóságnak, a méretezhetőségnek és a magas rendelkezésreállásnak. Ezen túl a DSI a következőket támogatja: egységes keretrendszer, folyamatszintű megközelítés, többszálú biztonság, rugalmas biztonsági szabályzat, átlátszó kulcskezelési réteg – mindezek mellett a teljesítményt csak kis mértékben fogja vissza.
A telep kezelése egyetlen gépről, a biztonságért felelős kiszolgálóról végezhető el. Ez a kiszolgáló felel a rendszerben található összes biztonsági alkotórészért, és ez felelős az osztott biztonsági szabályzat betartásáért is. A biztonsági politikában történt változásokat a kiszolgáló szétküldi a biztonsági kezelők felé, és ezáltal rugalmas környezetet alakít ki.
A biztonsági kezelők gondoskodnak helyileg a telep összes csomópontján a szabályok betartásáról és módosításáról. A biztonsági kezelők biztonsággal összefüggő adatokat csak a biztonsági kiszolgálóval cserélnek.

Osztott jogosultsági szerkezet

Egy telep hatékony jogosultsági rendszerének a megtervezése bonyolult feladat. Rengeteg tényezőtől függ a hozzáférési jogosultságok kiosztása, mivel az igénylők és az erőforrások a telepben bárhol elhelyezkedhetnek. Hogy leegyszerűsítsük



2. ábra Osztott jogosultság-ellenőrző rendszer

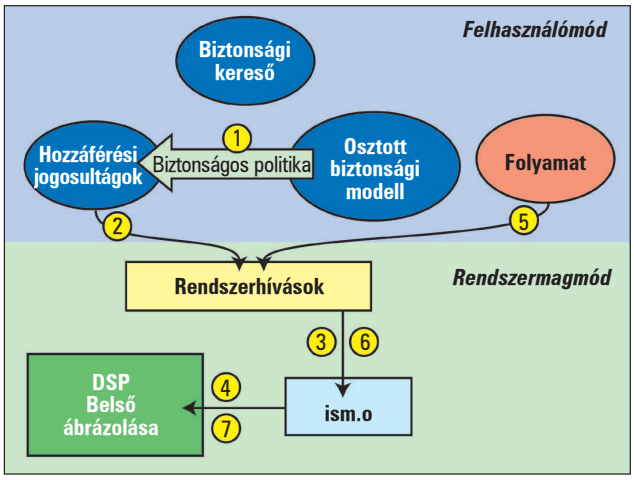
a kapcsolatokat, a hozzáférési jogosultságokat két szinten kezeljük. Helyi szinten az igénylő és az erőforrás egyetlen csomóponton található, míg távoli szinten az igénylő és az erőforrás különböző csomóponton helyezkedik el. Helyi szinten a jogosultságok az adott igénylői (`SSID`) és erőforrások erőforrási (`TSID`) biztonsági azonosítóihoz tartozó függvénynek megfelelően kerülnek elbírálásra (1. ábra). Ez a kiértékelés a Flask-szerkezet szerint történik:

```
HozzÆfØrØs = F ggvØny(SSID, TSID)
```

A Flask-szerkezet megoldást jelent egy csomópontra épülő feldolgozás esetében. Ha a gépek egy telepbe vannak összefűzve, a biztonság megoldása még bonyolultabbá válik. Ebben az esetben a Flask-szerkezetet a telep távolihozzáférés-modellre bővítjük (*2. ábra*).
Új kapcsolóinak egyike a biztonsági csomópontazonosító (`SnID`), vagyis egy olyan azonosító, ami a csomópontot határozza meg. Így a hozzáférési jogosultságok nemcsak az igénylőtől és az erőforrástól függnek, hanem a kérdéses csomóponttól is.
Az osztott jogosultsági rendszer szerkezete a 2. ábrán látható. A megfelelő szerkezet kiértékelése ennek megfelelően történik:

```
HozzÆfØrØs = F ggvØny(SnID2, SID2, SnID1, SID1)
```

Az osztott rendszer fontos része a hálózat, ami a csomópontokat köti össze. Ahhoz, hogy a hozzáférési jogosultságokhoz tartozó függvényeket a telepben alkalmazni tudjuk, a biztonsági azonosítókat a csomópontban észrevétlenül kell továbbítani. Osztott jogosultsági rendszerünk mintapéldányát a Linux-rendszermagban próbáljuk ki, ami az átlátszóságot és a jobb teljesítményt biztosítja nekünk.
A harmadik ábrán láthatjuk, hogyan működik az osztott biztonsági rendszer. A biztonsági kiszolgáló felelős a biztonsági szabályzatnak a bővítményhez történő továbbításáért. A kiszolgáló feladata az is, hogy minden egyes csomóponthoz egyedi biztonsági azonosítót rendeljen (1.). Tételezzük fel, hogy igénylő2 `SnID2` csomópontban megpróbál elérni egy erőforrást (egy fájlt) az `SnID1` csomóponton (2.). Ebben az



3. ábra Jogosultság-ellenőrzés egy csomóponton

esetben igénylő2-nek először jogosultságokat kell szereznie a helyi kapcsolattartási erőforrásokhoz, és hozzá kell jutnia két azonosítóhoz (`SnID2`, `SID2`), amiket azután továbbít a másik csomópontnak (4.). Ezeknek az azonosítóknak az érvényességét `SnID1`-es csomópont is ellenőrzi. Amennyiben a hozzáférés megadható, az üzenet továbbítható az 1-es folyamatnak (5.). Ekkor az 1-es folyamat fog hozzáférésért folyamodni a 2-es folyamat nevében (6.).

Hozzáférési jogosultságok a csomóponton

Ez a rész egy kicsit nagyobb részletességgel magyarázza el, mi történik a telep egyetlen csomópontján. A hozzáférési jogosultságok ellenőrzése a telep minden csomópontján két lépésben történik (*3. ábra*).

1. ***Rendszermagszinten:*** ez a szint felelős az elbírálásért és a döntések betarttatásáért, amiket külön felelősségként kezel. A rendszermag karbantartja a biztonsági politikát, és ennek alapján hozza meg döntéseit. A biztonsági politikát a biztonsági kiszolgáló szolgáltatja, és a gyors elérhetőség érdekében a helyi memóriában tárolódik.
2. ***Felhasználói szint:*** megannyi feladata közül (3. ábra) az egyik az osztott biztonsági politikának (1.) a biztonsági szerepkörök tárolóhelyéről a rendszermaghoz történő eljuttatása. Miután a szerepköröket összeállította, olyan formában továbbítja őket a rendszermaghoz, hogy az már könnyedén feldolgozhatja (2., 3., 4.). A rendszermagban keletkező riasztások is erről a szintről jutnak vissza a biztonsági kezelőhöz, ami a naplózó és elemző szolgáltatásokhoz továbbítja, illetve szükség esetén egy biztonságos közvetítő csatornán keresztül a biztonsági kiszolgálóhoz is továbbítja őket.

Mindkét szintet a helyi biztonsági kezelő (SM) indítja el és figyeli minden egyes csomóponton. Az SM hangolja össze a DSI alatt futó egyéb olyan szolgáltatásokkal is, amelyekkel kapcsolatba kell lépniük. Ha egy felhasználói folyamat a rendszer-erőforráshoz szeretne hozzáférni (5.), a rendszerhívás továbbítódik a DSM felé (6.), ahol végül megszületik a döntés a DSP belső ábrázolásának megfelelően (7.).

Címkék

Minden igénylőnek és erőforrásnak rendelkeznie kell azonosítóval. Mivel a biztonsági bővítmény már futási időben betöltődik, kétfajta igénylőcímzés között teszünk különbséget:

1. A bővítmény betöltődése előtt az igénylők és erőforrások nem kapnak címkét a rendszerben. A bővítmény betöltődése során minden futó folyamatot megvizsgál és címkével lát el.
2. Ha egy folyamat a bővítmény betöltődése után jön létre, a biztonsági kapcsok végzik el a címkézést.

Mivel a Linux a folyamatleírókat és a rendszermag-üzemmódú folyamatvermet egyetlen 8 KB-s memóriaterületen tárolja, az igénylők címkéinek szükségtelen külön további memóriaterületeket lefoglalnunk.
A többi címke futásidőben rendelődik hozzá az erőforrásokhoz, ennek következtében a bővítmény ellenőrzi a címke meglétét. Ha hiányzik a címke, újat hoz létre.

Hálózati címkék

Mivel a telepben egy igénylő a más csomópontokon lévő erőforrásokhoz is hozzáférhet (2. ábra), az ilyenfajta hozzáféréseket is ellenőrizni kell. Ha egy folyamat az egyik csomóponton egy másik csomóponton lévő erőforrást akar elérni, először a helyi kapcsolattartó eszközök (foglalat, hálózati eszköz) eléréséhez való jogosultságokat kell ellenőrizni. Ha a helyi hozzáférés rendelkezésre áll, az üzenet továbbítható a másik csomópontra.
Ahhoz, hogy a küldő igénylőt azonosítani lehessen, az IP-csomaghoz mellékelni kell a küldő csomópont és az igénylő biztonsági azonosítóját. Ennek a megvalósítására az adatcseréhez az IP-t használjuk fel. Az IP-protokollveremben a kapcsokon keresztül új értékek adódnak az IP-fejléchez. A fogadóoldalon ezeket az adatokat (a csomópont biztonsági azonosítóját és az igénylő biztonsági azonosítóját) az ottani IP-protokollveremben található kapcsok hálózati biztonsági azonosítóvá alakítják (NSID). A kiértékelés az alábbi módon történik:

```
NSID = F ggvØny (SnID, SID).
```

Ezt a függvényt a biztonsági kiszolgáló egy átalakítótábla formájában is szolgáltathatja (jelenleg egy egyszerű matematikai függvény használatos erre a célra). A fogadóoldal tehát úgy állítja elő a hálózati biztonsági azonosítót, hogy az átalakítótáblában megkeresi az `SnID`-hez és `SID`-hez tartozó bejegyzést. Ezután az `NSID` már helyi azonosítóként használható az erőforrások elérésekor.

Kísérletezés a DSM-mel

Több lépést is végre kell hajtanod, hogy le tudd fordítani és futtatni tudd a DSM-et. A példában feltételezzük, hogy a te rendszereden is Red Hat 7.2 fut 2.4.17-es Linux-rendszermaggal (a ➲ http://kernel.org-ról).
Az elvégzendő lépések a következők (a későbbiekben részletesebben tárgyaljuk őket):

- Alkalmazd rá az LSM-foltot a 2.4.17-es rendszermagra.
- Módosítsd a rendszermag beállításait, és fordítsd újra ezekkel az új beállításokkal.
- Frissítsd a rendszerindítási beállításokat a */etc/lilo.conf* fájlban.
- Indítsd újra a számítógépet az új rendszermaggal.
- Fordítsd le és futtasd a biztonsági bővítményt.
- Végezz el néhány kísérletet, hogy lásd, a bővítmény megfelelően működik-e.

A rendszermag újrafordítása az LSM beállításaival

Az osztott biztonsági bővítmény az LSM hátterére épül (ami a rendszermaghoz kötődő kapcsokból tevődik össze), és a biz-

tonsági folt az LSM weboldaláról tölthető le. Az oldal több különböző foltot is tartalmaz; neked a rendszermagodhoz illő változatot kell letöltened, ami ebben az esetben a 2.4.17-es. A következő lépéseket hajtsd végre a folt telepítéséhez. Először töltsd le az ***lsm-full-2002_01_15-2.4.17.patch.gz*** fájlt a */usr/src* könyvtárba, és csomagold ki a foltot:

```
% gunzip lsm-full-2002_01_15-2.4.17.patch.gz
```

Ezt követően lépj be a rendszermag forrásához, és alkalmazd a foltot:

```
% cd /usr/src/linux
% patch -p1 <
/usr/src/lsm-full-2002_01_15-2.4.17.patch
```

Miután ez megvan, állítsd be a rendszermagot, hogy támogassa az új bővítményt. A következő beállításokon kell változtatni:

- `CODE MATURITY LEVEL`: fejlesztés alatt álló és befejezetlen kódok, eszközmeghajtók mutatása. Ez elengedhetetlen.
- `LOADABLE MODULE SUPPORT`: ezt a legjobb kikapcsolni, hogy az eltérő változatszámokkal ne legyenek gondjaink.
- `NETWORKING OPTIONS`: a hálózati csomagszűrést feltétlenül engedélyezzük, mivel az IP-veremben található kapcsok csak így működnek. A rendszermagszintű HTTP-gyorsítást állítsuk `m`-re, így a rendszermagban a `tcp_sync_mss` beállítás elérhető lesz.

1. táblázat **Teljesítménypróbák IP-csomagmódosítással**

	Linux 2.4.17	*Linux 2.4.17 DSM-mel*	*Túlterhelés %*
Fork	167,00	169,00	+1,20%
UDP helyi hozzáférés (üzenetküldés)	16,3888	19,70	+20,00%
UDP távoli hozzáférés (hurokeszköz)	133,44	173,88	+30,00%

2. táblázat **Teljesítménypróbák IP-csomagmódosítás nélkül**

	Linux 2.4.17	*Linux 2.4.17 DSM-mel*	*Túlterhelés %*
UDP helyi hozzáférés (üzenetküldés)	16,388	17,084	+4,2%
UDP távoli hozzáférés (hurokeszköz)	133,44	140,64	+5,4%

- `SECURITY OPTIONS`: a ***capabilities support***-ot állítsuk `m`-re, az ***IP networking support***-ot `n`-re, az ***NSA SELinux Support***-ot `m`-re, az ***NSA SELinux Development Module***-t `y`-ra, az ***NSA SELinux MLS policy (EXPERIMENTAL)***-t `n`-re, az ***LSM port of Openwall (EXPERIMENTAL)***-t `n`-re, és a ***Domain and Type Enforcement (EXPERIMENTAL)***-t `n`-re.

Ha ezzel megvagy, egyszerűen fordítsd újra a rendszermagot és az új bővítményeket, majd futtasd a `lilo` parancsot, ahogy szoktad.

A DSM telepítése és használata

Mivel jelenleg az összes DSI-bővítmény még nincs működőképes állapotban, létrehoztunk néhány olyan próbaprogramot, amik utánozzák a hiányzó részeket, például a biztonsági szabályzat betöltését vagy a riasztások fogadását.

A bővítményt használat előtt először is be kell töltenünk a rendszermagba:

```
% /sbin/insmod lsm.o
```

Ezután a szabályzatot biztosítani kell a biztonsági bővítménynek. Erre a feladatra a biztonsági fájl egy egyszerű szöveges fájl lesz négy mezővel: ***forrás biztonsági azonosító***; ***cél biztonsági azonosító***; ***osztály*** (jelenleg csak háromféle osztály van elkészítve: a folyamatok sokszorosításáért, a foglalatokért és a hálózatokért felelős); végül pedig az ***engedély***.

```
1 1 1 0x01
1 1 2 0x07
1 1 3 0x01
```

A szabályzatot a próbaprogramunkkal tölthetjük be, amit ***UpdatePolicy***-nak hívunk:

```
% UpdatePolicy szabÆlyzat_fÆjl
```

A riasztásokat egy másik próbaprogrammal foghatjuk el, a `CheckAlarm`-mal. A programot így indíthatjuk el:

```
% CheckAlarm
```

Egy folyamat alapértelmezett azonosítóján úgy változtathatunk, ha az ELF formájú fájl kitöltő részében a legelső bájtot megváltoztatjuk (ez a fájl 8. bájtja).

A kísérleti rendszer

Háromfajta kipróbálási típust hajtottunk végre, hogy előzetesen meg tudjuk állapítani a biztonsági bővítmény teljesítményét. A legelső próba a folyamatok szétágaztatásán és létrehozásán alapul (`fork`), a második az UDP helyi, a harmadik az UDP távoli elérésén. Az UDP-próbát IP-fejlécmódosítással és anélkül is elvégeztük, hogy lássuk, mennyire erőforrás-igényes ez a módosítás.
A kipróbálást egy Pentium III-as 650 MHz-es gépen végeztük el, amiben 256 MB RAM volt. Az alábbi próbamódszerekkel dolgoztunk:

1. ***Folyamatok létrehozása***: megmérjük, mennyi idő szükséges egy új folyamat létrehozásához, ami nem tesz mást, csak azonnal befejezi a futását. A szülőfolyamat százezerszer futtatja le a `fork` és `wait` hívásokat.
2. ***UDP helyi elérés***: megmérjük, mennyi időbe telik egy UDP-üzenet küldése. A folyamat egy ciklusból ötszázezer UDP-üzenetet küld el. A küldő folyamat nem ellenőrzi, hogy a csomag valóban el lett-e küldve, visszajelzésre sem vár, hogy a címzett megkapta-e. Ebben az esetben nem lényeges, hogy a kiszolgálón telepítve van-e a DSM vagy sem.
3. ***UDP távoli hozzáférési próba***: leméri, hogy egy folyamatnak mennyi idejébe kerül egy UDP-üzenet elküldése és a kiszolgálótól egy UDP-üzenet fogadása. Ez a próba százezer UDP-üzenetet fogad és küld egy ciklusból. Az ügyfélprogram – amint visszaigazolást kapott az előző csomagról – újabb csomagot küld. Ebben az esetben lényeges, hogy a kiszol-

gálon fusson a DSM-szolgáltatás, hogy a fogadóoldalon a jogosultságokat megfelelően ellenőrizze. Próbánkban a második kiszolgáló egy Pentium II-es 300 MHz-es asztali gép 128 MB RAM-mal.

A teljesítménypróba eredménye

A próbák eredményei az *1.* és a *2. táblázatban* láthatók. A számok mikroszekundumban vannak megadva.

1. ***Folyamatok létrehozása***: 1,2 százalékos teljesítménycsökkenéssel kell számolnunk, mivel a rendszernek a `fork` művelet elvégzésekor ellenőriznie kell a hozzáférési jogosultságokat, és a keletkezett gyermekfolyamatokat megfelelően fel kell címkéznie.
2. ***UDP helyi elérés***: a DSM-bővítménnyel és az anélkül végzett próba során a teljesítménykülönbség húsz százalékos. Ebben az esetben a rendszernek el kell végeznie a jogosultságellenőrzést a foglalatokon, el kell küldenie az üzenetet és minden üzenethez az `sk_buff` címkemellékletet, valamint meg kell címkéznie az IP-csomagokat. Ha az IP-csomagokat nem módosítjuk (*2. táblázat*), a teljesítménykülönbség 4,2 százalékra csökken.
3. ***UDP távoli hozzáférési próba***: az átlagos teljesítménykülönbség a DSM-bővítménnyel és anélkül harminc százalék. Ebben az esetben a túlterhelést a következők jelentik: a foglalatokon el kell végezni a jogosultságok ellenőrzését, az `sk_buff`-hoz hozzá kell rendelni egy címkét, a biztonsági adatokat kapcsolni kell az IP-csomaghoz, a biztonsági adatokat le kell kérdezni a fogadóoldalon, a hálózati biztonsági azonosítót hozzá kell adni az `sk_buff`-hoz, el kell végezni az `sk_buff`-hoz tartozó jogosultságok ellenőrzését, és a foglalaton újra el kell végezni a biztonsági ellenőrzést, majd ugyanezt meg kell ismételni a másik oldalon is. Ha az IP-csomagot nem módosítjuk (2. táblázat), a teljesítménykülönbség 5,4 százalékra mérséklődik.

Mindkét UDP-próbában a legnagyobb plusz terhelést az IP-csomagok módosítása jelentette. A biztonsági bővítmény csak viszonylag kevés plusz terhelést okozott.

Folyamatban lévő munkák

A DSM-mel kapcsolatos terveink és folyamatban lévő munkáink:

- A jogosultságokat ellenőrző szétosztott keretrendszer teljes megvalósítása.
- A kód gyorsabbá tétele.
- Biztosítani a kiszolgáló erőforrásainak, hogy egy ügyfél nevében tevékenykedhessen.
- A biztonsági adatok védelme.
- Áthelyezni a biztonsági adatok szállítását egy alacsonyabb hálózati rétegbe.
- A telep biztonságának ellenőrzése különféle támadási módszerek esetében.

Összegzés

Mint korábban részleteztem, a DSM a DSI-szerkezet része. Eddig csak alapvető DSM-megvalósítással rendelkeztünk. A teljesítménypróbákat úgy kell tekinteni, hogy egyetlen biztonsági eljárást sem tettünk hatékonyabbá.
Az IP-csomagok feldolgozását kissé felgyorsítottuk. Az elsődleges eredmények jelentős teljesítménynövekedést mutattak. A helyi UDP-kezelés IP-csomagmódosítással húszról nyolc százalékra csökkent (1. tábla), és ehhez hasonlóan a távoli UDP-hozzáférés is harminc százalékos különbségről 14 százalékosra gyorsult. Ezek az eredmények ígéretesek, és vannak további ötleteink arra nézvést, hogyan gyorsíthatnánk fel egyes folyamatokat még jobban. Mindazonáltal az eredmények jól mutatják, hogy milyen kihívásokkal kell szembenéznie egy osztott rendszerek biztonságával foglalkozó fejlesztőnek. A rendszert valódi környezetben is kipróbáltuk, kísérletet tettünk néhány tártúlcsordulásos támadással, és ez megmutatta, hogy a jelenlegi rendszer ellenáll az ilyen támadási próbálkozásoknak. Végül megjegyeznénk, hogy minden tőlünk telhetőt megtettünk, hogy a lehető legjobban leírjuk a DSM működését a rendelkezésünkre álló elég szűkre szabott hely keretein belül. Amennyiben további részletekre vagy kíváncsi, nyugodtan keress meg minket. Reméljük, ki fogod próbálni a DSM forráskódját és a hozzátartozó próbaprogramokat, és a tapasztalataidat megírod nekünk. A teljes forráskód a CD-melléklet Magazin/DSM könyvtárában megtalálható.

Köszönetnyilvánítás

Köszönjük ***David Gordon***-nak és ***Philippe Veillette***-nek, a Sherbrooke Egyetem Számítástechnikai Tanszék munkatársainak a DSI projekten belül a DSM-területén végzett munkásságát.

Linux Journal 2002. október, 102. szám

Miroslaw Zakrzewski
kutató az Open Systems Labnél az Ericsson kutatóközpontban: az osztott biztonsági bővítmény fő fejlesztője. Elérhető a Miroslaw.Zakrzewski@Ericsson.ca címen.

Ibrahim Haddad
az Ericsson Corporate Unit of Research kutatója Montreálban, Kanadában. Címe: Ibrahim.Haddad@Ericsson.ca.

Kapcsolódó címek

CIPSO és FIPS-188 szabványok
➲ http://csrc.nist.gov/publications/fips/fips188.html

DSI Linux Journal Article
➲ http://www.linuxjournal.com/article.php?sid=6053

DSI Reference
➲ http://www.risq.ericsson.ca/dsi

A DSM az ottawai Linux Symposiumon
➲ http://www.risq.ericsson.ca/dsi/access_control_ols_paper.pdf

DSM-bemutató
➲ http://www.risq.ericsson.ca/dsi/AccessControl_OLS.pdf

Flask-szerkezet
➲ http://www.cs.utah.edu/flux/fluke/html/flask.html

Linux-rendszermag ➲ http://www.kernel.org

LSM ➲ http://lsm.immunix.org

SelOpt Patch
➲ http://www.intercode.com.au/jmorris/selopt/old