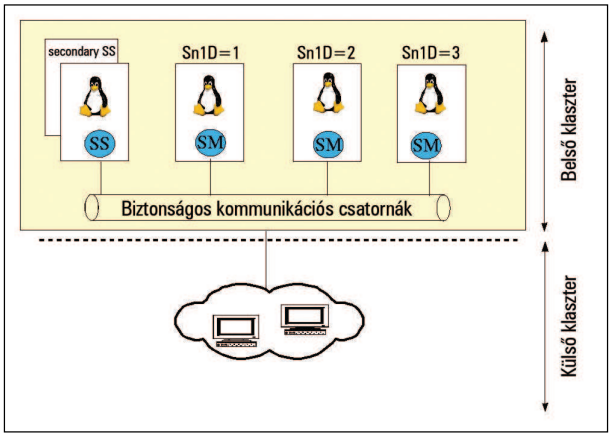
Virtuális biztonsági zónák létrehozása egy Linux fürtön belül

Az osztott biztonsági háttér lehetővé teszi, hogy egy Linux fürtön belül különálló virtuális biztonsági zónákat alakítsunk ki. Ez az írás ezek megvalósításáról és használatáról ad áttekintést.

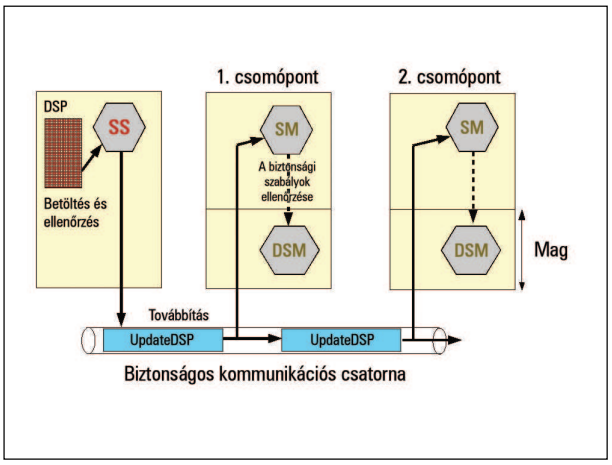
Egyre több projektben használják a ***Linuxot*** és más nyílt forrású programokat fürtözött számítógéprendszerek alapvető építőelemeként. A példák sora a mozifilmek gyártása során használt vizuális effektusoka előállító, óriási tömegű számítást végző fürtöktől a következő generációs telekommunikációs kiszolgálótelepekig terjed.
A technika fejlődésével és a méretek növekedésével párhuzamosan egyre gyakrabban fordul elő az is, hogy különféle szempontok – például a gazdaságosság, a kezelhetőség vagy a rugalmasság fokozása végett – célszerű ugyanazon a fürtön több különböző alkalmazást is futtatni. Jó példa erre a távközlés világában alkalmazott úgynevezett ***carrier-grade*** osztályú kiszolgálótelepek, amelyek erőforrásait eleve több különböző operátor között osztják meg. A közös használat ilyenkor azt jelenti, hogy az operátorok osztoznak a fürt teljes infrastruktúráján, ezeket használva esetleg egészen eltérő szolgáltatásokat nyújtanak az ügyfeleik számára, ugyanakkor a programjaikat és adataikat – nyilvánvaló gazdasági és biztonsági megfontolások miatt – nem szeretnék mások számára is elérhetővé tenni. Ezekben az esetekben a fürtök rendszergazdái sem férnek hozzá a programok forráskódjaihoz, és a biztonsági eljárásokat sem lehet a forráskód szintjén kötelezővé tenni. Szükség van tehát egy olyan biztonsági háttérre, amely biztosítja, hogy egy adott alkalmazás erőforrásait ne használhassák mások, illetve az adott szoftver működését ne zavarják meg a fürt más elemei.
Az ilyen helyzetekre az úgynevezett osztott biztonsági infrastruktúra (***DSI***, ***Distributed Security Infrastructure***) megoldást jelenthet. Ez a ***carrier-grade*** osztályú ***Linux*** telepeknél a fürt virtuális alfürtökre osztásával kísérli meg egy összefüggő biztonsági keretrendszer kialakítását. Biztosítja az alegységek közti kapcsolatok ellenőrzését, illetve a rendszergazdák által megadott szabályok alapján történő korlátozását. Annak ellenére, hogy a fejlesztések még csak két éve kezdődtek el, úgy gondoljuk, a ***DSI*** rendkívül hasznos eszköz lesz a fürtök rendszergazdáinak kezében. Ebben a cikkben bemutatjuk, hogy milyen módon használhatjuk a ***DSI***-t a ***Linux*** telepeken belüli virtuális biztonsági zónák kialakítására.



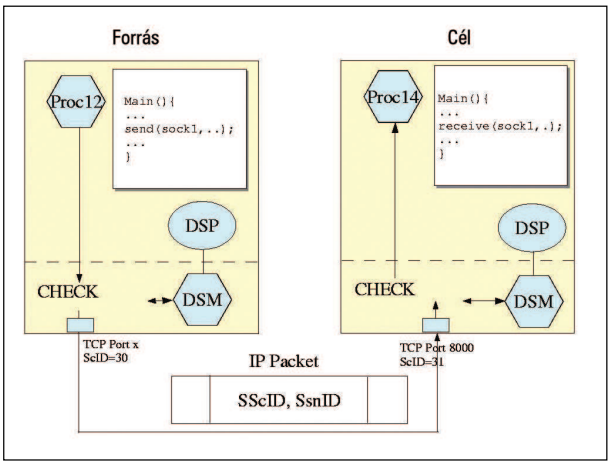
1. ábra A DSI felépítése

A DSI felépítése és eszközei

Először vizsgáljuk meg röviden a ***DSI*** felépítését. A ***DSI*** egy ***biztonsági kiszolgálóból*** (***SS***, ***Security Server***) és csomópontonként egy-egy ***biztonság-vezérlőből*** (***SM***, ***Security Manager***) épül fel (*1. ábra*). A biztonsági kiszolgálóban összpontosul a fürt ellenőrzése. Ez gyűjti össze a biztonsági vezérlők felől érkező figyelmeztetéseket és riasztásokat, és ez propagálja a fürtön belül az egyedi biztonsági intézkedéseket. A másik oldalon a biztonsági-vezérlők felelősek a biztonsági intézkedések betartatásáért, de kizárólag a saját csomópontjukon belül. A biztonsági kiszolgáló és a biztonsági vezérlők közt titkosított és hitelesített csatornákon zajlanak az üzenetváltások, amelyek a ***CORBA*** eseménycsatorna felett működő ***SSL/TLS*** használatával kerülnek továbbításra.
A ***DSI*** biztonsági elvei a folyamatok szintjén kerültek megvalósításra, vagyis a rendszer egy adott folyamat bizonyos erőforráshoz kötődő hozzáférési jogosultságait ellenőrzi. A folyamatok azonosítására a ***biztonsági kontextus azonosító*** (***ScID***) és annak a csomópontnak az azonosítója (***SnID***) szolgál, amelyen a folyamat fut. Ez a két adat egyértelműen meghatározza a rendszer számára a kérdéses folyamatot.



2. ábra A DSP elterjesztése a fürtön belül



3. ábra Biztonságos távoli elérés ellenőrzés

Az ***SnID***-ket a ***DSI SetNodeID*** nevű eszköze osztja ki. Minden azonos biztonsági kontextusba tartozó folyamat ugyanazzal az ***ScID***-vel rendelkezik. Az ***ScID***-k hozzárendelése történhet önműködően. Ilyenkor a rendszer osztja ki ezeket a megadott ***DSP***-szabályok szerint (lásd lejjebb). Ugyanakkor oszthatunk ki azonosítókat a ***DSI SetSID*** eszközével is, amikor egy konkrétan megnevezett futtatható állományhoz rendelünk hozzá kontextust. Ez utóbbi lehetővé teszi maguknak a futtatható állományoknak a biztonsági kontextus szerinti csoportosítását, vagyis a szabályrendszer nem csak a már futó folyamatokra alkalmazható.

A DSP beállítófájlja

Egy biztonsági szabály meghatározása a ***DSI***-ben valamilyen engedély megadását vagy megtagadását jelenti egy ***SnID-ScID*** párra vonatkozóan. Ezek a szabályok az egész fürtre érvényesek. A könnyű kezelhetőség érdekében minden szabály egy a biztonsági kiszolgálón lévő központi ***XML*** fájlban tárolódik.

A ***DSI*** lehetőséget biztosít az egész fürtözött telep egyedi és egységes kezelésére, valamint a szabályok frissítésének és alkalmazásának önműködő végrehajtására. Amint az adminisztrátor módosít egy már létező szabályt vagy újat hoz létre az osztott biztonsági házirendben (***DSP***, ***Distributed Security Policy***), a ***DSP***-t a ***dsiUpdatePolicy*** eszköz segítségével be kell tölteni a biztonsági kiszolgálóra. Ezután a ***dsiUpdatePolicy*** egyezteti a ***DSP***-t és a ***DSP XML*** sémáját (szintaktikai ellenőrzés). Amennyiben a ***DSP*** megerősítést nyer, a biztonsági kiszolgáló a biztonsági csatornákon keresztül a fürt minden csomópontjához eljuttatja az új szabályokat. Végül minden biztonsági vezérlő a ***DSM*** hívásával rendszermag szinten érvényesíti ezeket a szabályokat (lásd *2. ábra*). A ***DSM*** az ***LSM*** rendszermag-foltra épül. Ennek részletes ismertetésére nincs ebben a cikkben lehetőség, de a hálózaton elérhető (➲ ***www.linuxjournal/article/7688***) részben találhatunk hivatkozásokat erre vonatkozólag.

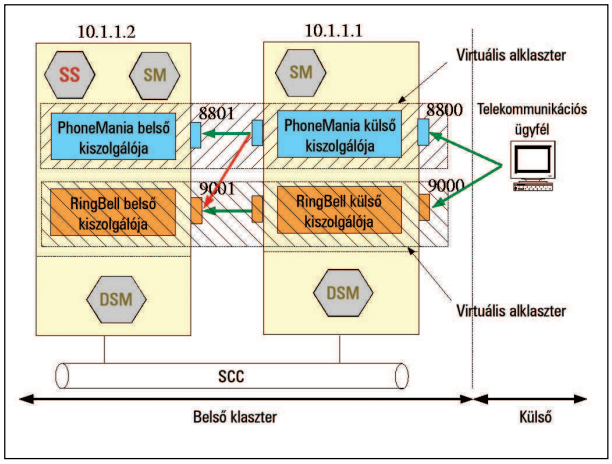
Osztott hozzáférés-ellenőrzés

A helyi erőforrásokhoz való hozzáférés ellenőrzése viszonylag egyszerű. A ***DSM***-modul megkapja a kérést kibocsátó folyamathoz tartozó ***ScID***-t és ***SnID***-t, majd ellenőrzi az érvényben lévő biztonsági szabályok között, hogy az rendelkezik-e a a megfelelő engedélyekkel. A ***DSI***-ben tulajdonképpen az az igazán eredeti, hogy megosztott módon végzi a hozzáférések ellenőrzését. Egyelőre még csak a foglalatok közti párbeszédet valósították meg. Ennek bemutatására egy olyan hozzáférés-ellenőrzési helyzetet mutatunk be, amikor a folyamat egy másik csomóponton lévő erőforrást próbál meg elérni (*3. ábra*):

- A hozzáférési kérést a helyi ***DSM*** fogadja, majd ellenőrzi, hogy a folyamatnak egyáltalán van-e joga a helyi foglalatokkal kapcsolatos rendszerhívások kibocsátására.
- Ezután a ***DSM*** a távoli csomópontnak elküldött minden IP-csomaghoz hozzáfűzi a kérő folyamathoz tartozó ***ScID*** és ***SnID*** azonosítókat.
- A fogadó csomóponton a távoli ***DSM*** a kérést indító folyamat az IP-csomagokból visszanyeri a ***ScID*** és ***SnID*** azonosítókat, majd ezeket használva ellenőrzi, hogy a kérő folyamat rendelkezik-e engedéllyel a célkerettel és a célkeretet birtokló folyamattal való adatcseréhez.
- Végül a távoli ***DSM*** helyben ellenőrzi, hogy a folyamat, amelyhez a megcélzott keret tartozik, fogadhat-e információt a kérést kezdeményező folyamattól.

A probléma

Ebben a részben részletesebben megvizsgálunk egy viszonylag egyszerű esetet, amelyen keresztül közérthetően bemutatható, hogy egy adott problémát hogyan lehet a ***DSI*** rendszer segítségével megoldani. Tegyük fel, hogy egy két csomópontból álló egységet szeretnénk megosztani két távközlési operátor között, akiket most ***PhoneMania*** és ***RingBell*** néven fogunk emlegetni, s akik a saját alkalmazásaikat futtatják a fürt csomópontjain. Mindketten telefonos reklám- és információszolgáltatást nyújtanak, amely abban merül ki, hogy a végfelhasználók felhívhatják a (***TelecomClient***-et használó) belépési pontokon lévő kiszolgálókat és lekérhetik onnan a megadott vállalatok árajánlatait. A belépési pontokon lévő kiszolgálók (***PhoneManiaEP*** és ***RingBellEP***) továbbítják a kérést a háttérben működő kiszolgálóknak (***PhoneManiaBE*** és ***RingBellBE***), amelyektől aztán megkapják az ajánlatokat és visszaküldik azokat a végfelhasználónak.



4. ábra Egy egyszerű távközlési eset

Ha valaki hozzászokott a fürtözött rendszerben való gondolkodáshoz, ezen a ponton a következő kérdés merül fel: hogyan védhetnénk meg egy ***PhoneMania*** alkalmazást attól, hogy a kéréseit tévedésből a ***RingBell*** háttérkiszolgálóinak továbbítsa? Ha semmilyen erre vonatkozó biztonsági rendszabályt nem léptetünk életbe, akkor a ***PhoneMania*** megtehetné ezt olyan esetekben, amikor a háttérkiszolgálója túlterhelt, vagy egyszerűen csak nem rendelkezik a kért információval. És akkor még nem is említettük az olyan súlyosabb eseteket, amikor mondjuk az előfizetők adatait akarják megszerezni illetéktelenek, vagy a versenytársnak akarnak szándékosan kárt okozni. Egy ilyen szituáció bemutatására minden szereplőt egyszerű ***UDP***-ügyfélként és -kiszolgálóként valósítottunk meg (*4. ábra*).
Az „eltévelyedés” forgatókönyve lépésenként a következő:

- `PhoneMania` és `RingBell` egy `munster` nevű csomóponton futtatják a háttérkiszolgálóikat:

```
[munster demo]$ ./RingBellBE -h 10.1.1.2 -p 9001
RINGBELL: bind on 10.1.1.2:9001
..
[munster demo]$ ./PhoneManiaBE -h 10.1.1.2
➥-p 8801
PHONEMANIA: bind on 10.1.1.2:8801
```

- Ahogy `PhoneMania` túlterheltté válik, úgy dönt, hogy `RingBell` erőforrásait fogja használni, így a `colby` csomóponton lévő `PhoneMania` belépési pont kiszolgálója (8800-as kapu) az ügyfeleitől érkező összes kérést a `RingBell` háttérkiszolgálóira irányítja (9001-es kapu):

```
[colby demo]$ ./PhoneManiaEP -h 10.1.1.1
➥-p 8800 -b 10.1.1.2 -r 9001
PHONEMANIA: bind on 10.1.1.1:8800
PHONEMANIA: connect on 10.1.1.2:9001
..
```

- Amikor egy ügyfél a `PhoneMania` belépési pontján (8800-as kapu) egy ajánlatot kér, `PhoneMania` valójában `RingBell` háttérkiszolgálóját használja a válasz megadására (9001-es kapu). Ennek pedig az lesz a – nem túl mulatságos – következménye, hogy `PhoneMania` kapja a pénzt `RingBell` erőforrásainak használatáért.

```
[colby demo]$ ./TelecomClient -h 10.1.1.1
➥-p 8800
Connecting to : 10.1.1.1:8800

Requesting quotation for Ericsson
Quote Ericsson
..
[munster demo]$
..
RINGBELL backend : processing quotation request
for Ericsson

RINGBELL backend : quotation for Ericsson is 83
Quote Ericsson
```

Ennek megelőzésére az egyik lehetséges megoldás az, ha a ***DSI*** segítségével a megosztott fürtöt biztonsági szempontból különálló egységekre, alfürtökre osztjuk fel. A következőkben lépésenként bemutatjuk, hogyan használható a ***DSI*** ennek a gyakorlati megvalósítására.

A DSI telepítése és beállítása

Először is a fürt minden csomópontjára telepítenünk kel a ***DSI***-t. Miután a ***SourceForge*** honlapjáról letöltöttük a ***DSI tar***-csomagját, le kell fordítanunk a saját gépünkön, mivel a program a szabványos fordítóeljárást alkalmazza.
A ***SourceForge*** oldalán található ***DSI***-leírásban részletesen ismertetjük a ***DSI*** fordításának és telepítésének lépéseit.
Minden csomóponton futtatni kell a ***Security Managert***, ami a kétcsomópontos fürtünk esetében ez azt jelenti, hogy a program a `colby` és `munster` csomópontokon fog futni:

```
[colby]$ cd ~/dsi
[colby]$ source dsi_setup.sh
[colby]$ ~/dsi/bin/dsiSecManager
```

Az egyszerűség kedvéért a `colby` a biztonsági kiszolgáló szerepét is betölti:

```
[colby]$ cd ~/dsi
[colby]$ source dsi_setup.sh
[colby]$ ~/dsi/bin/dsiSecServer
```

A biztonsági kiszolgáló és a biztonság-vezérlők egymással a ***CORBA*** eseménycsatornák használatával tartják a kapcsolatot.
Minden csomóponton betöltjük a ***DSI*** redszermagmodulját, a ***DSM***-et, amellyel a rendszermag szintjén biztosítható a biztonsági intézkedések betartása:

```
$ cd ~/dsi/lsm
$ su root
Password:
# ./load
# /sbin/lsmod
```

```
Module    Size  Used by    Not tainted
dsm      36332   0 (unused)
...
```

Ezután beállítjuk a ***DSI***-t külön IP-cím megadásával az egyes csomópontok számára a biztonságos és nem biztonságos adatkapcsolat számára. Ehhez egy `DciInit` nevű eszközt írtunk, amelynek használatáról és a ***dci_policy.conf*** fájl felépítéséről a ***SourceForge*** honlapján található ***DCI***-leírásban olvashatunk részletesebben:

```
$ cd ~/dsi/user/tools
$ ./DciInit ~/dsi/etc/dci_policy.conf
```

A megoldás: virtuális alfürtök létrehozása

Ahhoz, hogy különálló virtuális alfürtöket hozhassunk létre, lényegében külön ***ScID*** azonosítókat kell létrehoznunk a ***PhoneMania*** (a példánkban `ScID=10`) és a ***RingBell*** (`ScID=20`) erőforrásai számára. Ezután új szabályt léptetünk érvénybe a ***DSP***-ben, amely alapján a rendszer visszautasít bármilyen kapcsolatkérést, ami az `ScID=10` érték által meghatározott zónából az `ScID=20` övezetbe érkezik, illetve fordítva. Az egyes operátorokhoz tartozó erőforrások különálló csoportokba szervezésével és mindennemű köztük létrejövő kapcsolat tiltásával valójában a fürt egy virtuális felosztását értük el. Ehhez rendszerfelügyeleti céllal az adminisztrátor létrehozhatna még egy újabb zónát az `ScID=30` értékkel, amely mindkét övezethez rendelkezne hozzáféréssel.
Először is, rendeljük hozzá minden csomópont minden bináris állományához a megfelelő ***ScID*** azonosítót (ehhez a `SetSID` eszközt használjuk):

```
$ ~/dsi/user/tools/SetSID PhoneManiaEP 10
➥Changing from SID 0 to SID 10
$ ~/dsi/user/tools/SetSID PhoneManiaBE 10
➥Changing from SID 0 to SID 10
$ ~/dsi/user/tools/SetSID RingBellEP 20
➥Changing from SID 0 to SID 20
$ ~/dsi/user/tools/SetSID RingBellBE 20
➥Changing from SID 0 to SID 20
$ ~/dsi/user/tools/ls_dsi .
PERMISSION      USER     GROUP    BSID   FILE
-rwxr-xr-x      lmcaxpr install 10     PhoneManiaBE
-rwxr-xr-x      lmcaxpr install 20     RingBellBE
-rwxr-xr-x      lmcaxpr install 10     PhoneManiaEP
-rwxr-xr-x      lmcaxpr install 20     RingBellEP
```

Amikor a ***DSM*** betöltődik, foganatosítja az alapértelmezésként engedélyezett biztonsági szabályokat. A fürt felosztásának megvalósításához szerkesztenünk kell a ***DSP*** fájlját (***~/dsi/etc/SampleDSP.xml***) és minden már létező biztonsági szabályt a saját szabályainkkal kell helyettesítenünk.
A `PhoneMania` foglalatai az `ScID=10` azonosítót kapják, a `RingBell` pedig az `ScID=20` értéket használja. A következő szabály az `ScID=10` értéket rendeli a `PhoneMania` belépési pontjának ***UDP*** foglalatához (8800-as kapu):

```
<class_SOCKET_INIT_rule>
  <protocol>UDP</protocol>
  <port>8800</port>
  <SnID>ALL</SnID>
  <ScID>10</ScID>
</class_SOCKET_INIT_rule>
```

Három hasonló szabályra van még szükségünk: egy a `PhoneMania` háttérkiszolgálójához, két másik pedig az `ScID=20` érték `RingBell`-hez való rendeléséhez.
Ezután a `PhoneMania` folyamatainak (forrás `ScID=10`) engedélyezzük, hogy üzeneteket hozzanak létre, küldjenek vagy fogadjanak a saját foglalataikon (vagyis az `ScID=10` azonosítójú célokon):

```
<class_SOCKET_rule>
  <sScID>10</sScID>
  <sSnID>ALL</sSnID>
  <tScID>10</tScID>
  <tSnID>ALL</tSnID>
  <allow>CREATE CONNECT LISTEN RECEIVE
      ➥SEND</allow>
</class_SOCKET_rule>
```

A `RingBell` számára hasonló szabályt hozunk létre. Természetesen az `ScID=10` és 20 közti adatcserét le kell tiltanunk. Ezt egyszerűen úgy tehetjük meg, hogy nem adunk engedélyeket ezen ScID azonosítók közti adatcseréhez:

```
<class_SOCKET_rule>
  <sScID>10</sScID>
  <sSnID>ALL</sSnID>
  <tScID>20</tScID>
  <tSnID>ALL</tSnID>
  <allow></allow>
</class_SOCKET_rule>
```

Hasonló szabályt hozunk létre az `ScID=20` forrású és `ScID=10` célazonosító között.
Bár a háttérkiszolgálók és egy adott operátorhoz tartozó belépési pontok kiszolgálói fizikailag a fürt különböző csomópontjain helyezkedhet el, ne feledjük, hogy megosztott fürtről van szó, tehát nem rendeljük az egyes csomópontokat egyértelműen, elválaszthatatlanul a `RingBell`-hez vagy a `PhoneMania`-hoz. Így a `PhoneMania` folyamatainak (forrás `ScID=10`) képesnek kell lennie egy másik `PhoneMania`-folyamattal (cél `ScID=10`) történő kapcsolatfelvételre is a hálózaton keresztül. Ugyanez természetesen a `RingBell`-re is igaz.

```
<class_NETWORK_rule>
  <sScID>10</sScID>
  <sSnID>ALL</sSnID>
  <tScID>10</tScID>
  <tSnID>ALL</tSnID>
  <deny>NETWORK_RECEIVE</deny>
</class_NETWORK_rule>
```

Végül a `PhoneMania` (`ScID=10`) és `RingBell` (`ScID=20`) folyamatokat rendszerint valamilyen parancsértelmezőből futtatjuk (az alapértelmezett `ScID=2`). Ez nyilván csak úgy lehet-

séges, ha engedélyezzük a kérdéses héj számára új folyamatok létrehozását. Ezt egy `transition` szabály segítségével tehetjük meg:

```
<class_TRANSITION_rule>
  <parent_ScID> 2 </parent_ScID>
  <SnID>ALL</SnID>
  <binary_ScID>10</binary_ScID>
  <new_ScID>10</new_ScID>
</class_TRANSITION_rule>
```

A `binary_ScID` a bináris állományhoz explicite hozzárendelt ***ScID*** azonosító. Emlékezzünk rá, hogy a `PhoneManiaBE` és `PhoneManiaEP` programokhoz a `SetSID` segítségével rendeltünk ***ScID*** azonosítót. A `new_ScID` a létrejövő új folyamathoz rendelt ***ScID***. Mivel a 8800-as és 8801-es foglalatok elérése csak az `ScID=10` azonosítójú folyamatok számára (vagyis a `PhoneMania` számára) engedélyezett, az új folyamathoz is az `ScID=10` azonosítót kell rendelnünk. Hasonló szabályt kell a `RingBell` számára is létrehoznunk. Mindössze ennyire van szükségünk a ***DSP*** létrehozásához, vagyis 12 egyszerű biztonsági szabályra, amelyek érvényesítése tökéletesen kizárja a fent vázolt problémát. A szabályrendszer leírása után természetesen az egész fürtre kiterjedően frissítenünk kell a biztonsági rendszabályokat. Ehhez valamennyi biztonsági kiszolgálónak egy `update` (frissítés) üzenetet kell elküldenünk:

```
[colby]$ cd ~/dsi/SS/test/demoSecOM
[colby]$ ./dsiUpdatePolicy ~/dsi/etc/
➥DSP.xml
```

A biztonsági kiszolgáló ennek hatására beolvassa a módosított ***DSP*** fájlt (***~/dsi/etc/DSP.xml***) és figyelmeztetést küld, ha valamilyen formai hibát észlel. Ha mindent rendben talált, önműködően elküldi a frissítéseket minden biztonsági vezérlő számára, tehát nincs szükség arra, hogy a csomópontokra egyenként bejelentkezve minden gépen kézzel frissítsük a biztonsági intézkedéseket leíró adatbázist, vagy hogy saját Perl alapú rendszerkezelő programot fejlesszünk erre a célra, amely legalább részben automatizálja a folyamatot. Minden magától működik, mint az álom. Ez különösen akkor jelent nagy előnyt, ha több ezer csomópontból álló teleppel dolgozunk, amelyben egyes csomópontok földrajzilag esetleg a világ másik végén is lehetnek. Gondoljunk például a hálószámítási (grid computing) módszerekre. Ott nemhogy gyakori, egyenesen természetes ez a helyzet.
Most, hogy rendszerünket bebiztosítottuk a forgalomirányítási eltévelyedések ellen, itt az ideje, hogy újra kipróbáljuk azt az esetet, amikor a `PhoneMania` kérést továbbít a `RingBell` háttérkiszolgálója felé:

```
[colby demo]$ ./TelecomClient  -h 10.1.1.1
➥-p 8800
Requesting quotation for Ericsson
Quote Ericsson
...
[colby demo]$ ./PhoneManiaEP -h 10.1.1.1
➥-p 8800 -b 10.1.1.2 -r 9001

PHONEMANIA: bind/connect on 10.1.1.1:8800 = 0
PHONEMANIA: bind/connect on 10.1.1.2:9001 = 0
Quote Ericsson

Quotation request received
...
[munster demo]$ ./RingBellBE -h 10.1.1.2 -p 9001
RINGBELL: bind on 10.1.1.2:9001
...
```

A másik csomóponton (`munster`) észleljük, hogy a `RingBell` háttérkiszolgálója többé már nem kezeli a `PhoneMania` kéréseit, pedig a `PhoneMania` engedély nélkül továbbra is a `RingBell`-re irányítja azokat. A */var/log/messages* fájlban a nem engedélyezett kérések nyomon követésének céljából lehetőség van a ***DSI*** által előállított naplók rögzítésére:

```
May  6 07:47:31 munster kernel: DSI-LSM MODULE -
➥dsi_sock_rcv_skb check permission sscid 10
➥ssnid 1 tscid 20
May  6 07:47:31 munster kernel: DSI-LSM MODULE
➥Error - dsi_sock_rcv_skb - No Permission
```

Összegzés

Egy gyakorlati megoldást mutattunk be a fürtök különböző felhasználók által használt alkalmazások közti biztonságos megosztására. A ***DSI*** projekt lehetővé teszi a felhasználók számára, hogy egyszerűen hozzanak létre elkülönülő biztonsági zónákat a telepen futó alkalmazások számára. Miután a ***DSI***-t feltelepítettük a telepre, az új alkalmazások számára létrehozandó új biztonsági zónák kialakítása a programokhoz tartozó megfelelő ***ScID*** azonosítók beállítására és a kapcsolódó szabályok ***DSP*** fájlban történő rögzítésére egyszerűsödik. A forráskód módosítására nincs szükség és valószínűleg lehetőség sincs rá.

Linux Journal 2004. október, 126. szám

Makan Pourzandi (makan.pourzandi@ericsson.ca) az Ericsson kanadai kutatóközpontjának nyílt rendszerekkel foglalkozó osztályán dolgozik. A kutatási területei közé a biztonság, a fürtözött számítások és az osztott programozás komponens alapú módszerei tartoznak. Doktori fokozatát 1995-ben Franciaországban, a lioni egyetemen szerezte meg a párhuzamos számítások tárgykörében írt dolgozatával.

Axelle Apvrille (axelle.apvrille@ericsson.ca) jelenleg az Ericsson kanadai kutatóközpontjának nyílt rendszerekkel foglalkozó osztályán dolgozik. Kutatási területei a kriptográfia, biztonsági protokollok és az osztott biztonság. Számítástechnikából 1996-ban szerzett diplomát a franciaországi Bordeaux-ban, az ENSEIRB egyetemen.