Osztott biztonsági rendszer Linux-telepeken

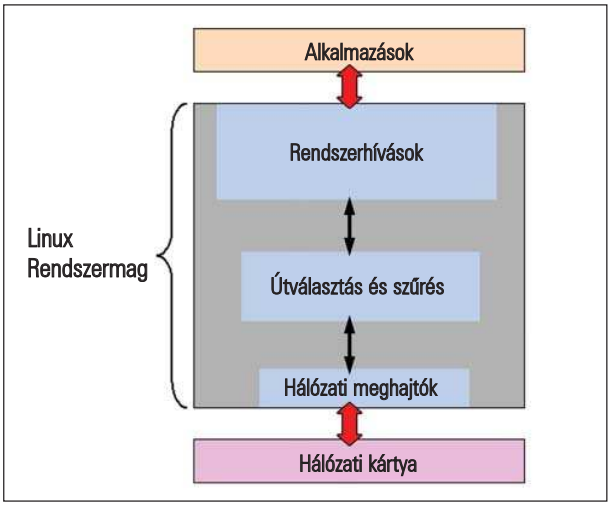
A cikkből megtudhatjuk, hogy milyen rendszermag-szintű eljárást használ az osztott biztonsági modul (DSM) a biztonsági információk IP-üzenetekbe való átlátszó-ágyazásához.

A jelen cikk az osztott biztonsági háttérről (***Distributed Security Infrastructure***, DSI) és az osztott biztonsági modulról (***Distributed Security Module***, DSM) korábban megjelent cikkek folytatása (lásd a ***Linuxvilág 2002 novemberi, 22.*** számában megjelent ***Az osztott linuxos biztonsági modell*** és a ***DSI: biztonságos Linux a távközlésben*** című cikkeket, valamint az eredeti angol nyelvű cikkeket a ***Linux Journal*** honlapján: ***„Linux Distributed Security Module"***, LJ, 2002 októberi szám, ➲ http://www.linuxjournal.com/article/6215, illetve a ***DSI: a New Architecture for Secure Carrier-Class Linux Clusters***, ➲ http://www.linuxjournal.com/article/6053). Jelen cikkben arra fókuszálunk, hogy egy osztott környezetben hogyan használjunk a DSM-ben IP beállításokat biztonsági információk küldésére a folyamatszintű biztonság megvalósításához. Kitérünk a hálózati átmeneti tár kezelésére, a kapcsok rendszermagbeli létrehozására, az IP beállításokra és az IP fejrész módosítására. Ezt követően a DSM hálózati kapcsait tárgyaljuk és bemutatjuk néhány előzetes teljesítményteszt eredményét.

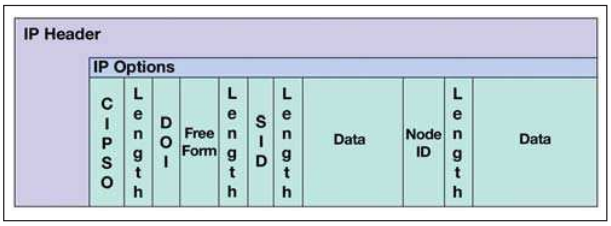
A DSI projekt

Az ***Ericsson Research*** nyílt rendszerekkel foglalkozó laboratóriuma a nyílt forrású DSI projektet azzal a céllal indította útjára, hogy egy olyan programalapú valós idejű kommunikációs rendszert fejlesszen ki, amely carrier-grade osztályú, Linux-alapú távközlési telepeken futtatható. Ezeknek a telepeknek bármilyen gép- vagy programhibától függetlenül folyamatos üzemmódban kell működniük, és lehetővé kell tenniük a karbantartók számára, hogy a programot, a berendezéseket, a rendszermagot vagy az alkalmazásokat működés közben frissítsék. Mindezt úgy, hogy az a kínált szolgáltatásokat ne befolyásolja, és ne legyen semmiféle leállás sem.

A DSI-t eredetileg az olyan carrier-grade jellemzők megvalósítására tervezték, mint a megbízhatóság, méretezhetőség, nagyfokú hozzáférhetőség és nagy teljesítmény. Ezeken felül számos más fontos jellemzővel is rendelkezik. Ilyen az egységes keretrendszer, a folyamatszintű megközelítés, és az, hogy támogatja az időosztásos biztonsági szabályokat, és a menet közben frissíthető biztonsági rendet egyaránt. A DSI fontos jellemzője a folyamatszintű hozzáférés-ellenőrzés. A jelenleg megvalósított biztonsági eljárások a felhasználói jogosultságokon alapulnak, és nem támogatják azokat a hitelesítéseket, amelyek ugyanazon felhasználóhoz tartozó két folyamat kölcsönhatására vonatkoznak akár távoli feldolgozóegységek által létrehozott folyamatok esetén. Távközlési alkalmazások esetén csak kevés felhasználó futtatja megszakítás nélkül hosszabb ideig ugyanazt az alkalmazást. A fenti megközelítés használata a különböző hálózati csomópontokon indított folyamatok mindegyikének ugyanazokat a biztonsági jogokat adja, amely ahhoz vezet, hogy a megosztott rendszeren számos művelet biztonsági ellenőrzés nélkül futhat. Az említett



1. kép A hálózati csomag útja a rendszermagban



2. kép Biztonsági beállítások az IP-fejrészben

1. lista sock_sendmsg()

```
sock_sendmsg
(struck socket *sock, struct msghdr *msg, int
➥size)
{
  int err;
  struct scm_cookie scm;

  err =
    security_ops->socket_ops->sendmsg(sock,
                                      ➥msg, size);
  if(err)
    return(err);
  ...

}
```

2. lista dsi_socket_sendmsg()

```
int dsi_socket_sendmsg(struct socket *sock,
                       ➥struct msghdr *msg, int
                       ➥size)
{
...

inode_security_t *isec;
struck sock sk;
struct ip_options *opt =  NULL;
int optlen = NSID_BASE_LEN + NSID_SSID_LEN +
             NSID_NODEID_LEN; //8 +_6 + 6
unsigned char optptr[optlen];

  ...

  sk = sock->sk;
  opt = sk->protinfo.af_inet.opt;
  dsi_options_fill (isec, optptr, optlen);
  dsi_ip_options_get(&opt, optptr, optlen);
  opt = xchg(&sk->protinfo.af_inet.opt, opt);

...
}
```

biztonsági rendszer alapegysége a felhasználó. A vivőszintű (***carrier-class***) alkalmazások esetén ez a felosztás nem megfelelő, finomabb megközelítésre van szükség, ezért támogatja a DSI is az egyedi folyamatot, mint a felosztás alapegységét.

Az osztott biztonsági modul (DSM)

A DSM a DSI központi összetevője, amely biztosítja a Linux telepen belül a kötelező érvényű hozzáférés-ellenőrzés megvalósítását. A DSM felelős a hozzáférés-ellenőrzés végrehajtásáért, és biztosítja, hogy a telep csomópontjain áthaladó IP-üzenetek el legyenek látva a küldő csomópont és folyamat biztonsági jellemzőit tartalmazó címkével. A DSM az LSM-kapcsokat használó Linux modulként került megvalósításra. A fejlesztés a 2.4.17-es Linux rendszermag használatával kezdődött egy megfelelő LSM-rendszermagfolt alkalmazásával. Ez a megvalósítás az IP-fej módosítását leíró CIPSO és FIPS 188 szabványokra épül.

A DSM megvalósításának fontos eleme az osztott jelleg. Egy telepen belüli csomópontról kiinduló hozzáférés-ellenőrzés vonatkozhat egy olyan erőforrásra, amely egy másik csomóponton helyezkedik el, ezért szükség van arra, hogy létezzen a telepen belüli csomópontok között a biztonsági információk átvitelének a lehetősége. A DSM osztott jellege biztosítja, hogy az erőforrásokhoz való hozzáférés biztonsági szempontból helyfüggetlen legyen.

A hálózati átmeneti tár kezelése

Röviden összefoglaljuk a hálózati átmeneti tár kezelését annak érdekében, hogy jobban megérthessük, hogyan ágyazódik a biztonsági információ a hálózati csomagba. Leírjuk azt is, hogy a rendszermag hogyan kezeli a hálózati tárat a felhasználói rétegtől a hardverrétegig és vissza.

Az *1. kép* a hálózati csomag útját mutatja a rendszermagban. A csomagkezelés két esete a bejövő, illetve kimenő csomagok kezelése. A kimenő csomagok kezelése az alkalmazói rétegtől kezdődően a következőképpen alakul: az alkalmazás előkészíti az adatot a hálózaton keresztül történő továbbításra; az alkalmazás a csomag küldésére vonatkozó rendszerhívást küld a rendszermag felé; a csomag `sk_buff` szerkezetben áthalad a rendszermag szűrő és útválasztó függvényein; végül a csomag a hálózati meghajtóhoz kerül, amely azt a hálózati kártyára (DMA) továbbítja.

A bejövő hálózati csomag útja a hálózati kártyától kezdődően oly módon indul, hogy a kártya begyűjti a saját vagy csoportos (***broadcast***) címével megcímzett csomagot; ezután beolvassa a hálózati memóriába, és egy megszakítást hoz létre. A hardveres megszakítás által kiváltott és a hálózati kártya meghajtóprogramjának részeként a rendszermagban elhelyezkedő megszakítás-kiszolgáló rutin lefoglal egy `sk_buff` területet, majd a hálózati kártya memóriájából (DMA) ebbe az átmeneti tárba mozgatja át az adatokat. Ezt követően a csomag a felsőbb rétegbeli feldolgozás céljából a processzor várakozási sorába kerül, a feldolgozás pedig akkor folytatódik, amikor a megszakítás engedélyezetté válik. Végül a csomagok keresztülhaladnak a szűrőkön és útválasztó függvényeken, és az alkalmazói réteg felé továbbítódnak.

Most, hogy már ismerjük az általános módszert, amivel a Linux rendszermag kezeli a hálózati átmeneti tárat, bemutatjuk, hogy mindezt hogyan használhatjuk fel a rendszermag biztonságának növelésére.

Megvizsgáljuk az IP útválasztó függvényekhez létrehozott kapcsokat, amelyek lehetővé teszik az IP-csomagok befolyásolását és növelik az IP-üzenetek biztonsági lehetőségeit.

Hálózati biztonsági kapcsok létrehozása

A biztonsági modul képes befolyásolni a biztonsági kapcsok megvalósításán alapuló útvonal kiválasztási döntéseket. Mivel az útválasztó kapcsok a rendszermagba érkező és

3. lista ip_options_compile ()

```
int
ip_options_compile (struct ip_options *opt,
                    ➥struct sk_buff *skb)
{
unsigned char *pp_ptr;
unsigned char *optptr;

  ...

  case IPOPT_CIPSO:

    if(security_ops->ip_ops->decode_options(skb,
                              ➥optptr, &pp_ptr)
      goto error;
      break;
  ...
}
```

4. lista sock_queue_rcv_skb ()

```
int
sock_queue_rcv_skb (struct sock *sk,
                    ➥struct sk_buff *skb)
{
int err=0;

  ...

  err=security_ops->socket_ops->sock_rcv_skb
                                    ➥(sk,skb);

  if(err)
    return (err);
  ...
}
```

onnan kimenő csomagok esetén normál rendszermag-függvényként futnak, ezek programozásánál fel kell idéznünk néhány dolgot.
A függvényt regisztráló modulnak rögzítenie kell a függvény kapcson belüli prioritását. A hálózati szűrő kapcsok a rendszermag-kódjából a prioritásuknak megfelelő sorrendben kerülnek meghívásra.
A felhasználói függvények szabadon módosíthatják az IP-csomagokat, és az alábbiak közül kell egy értéket visszaadniuk a hálózati kód számára, hogy az eldönthesse, mi a teendő a csomaggal:

1. NF_ACCEPT: nincs semmi tennivaló, a csomagot átengedi a hálózati vermen.
2. NF_DROP: a csomag eldobása, nincs további feldolgozás.
3. NF_STOLEN: a csomag átvéve, nincs további feldolgozás.
4. NF_QUEUE: a csomag sorbaállítása felhasználói kezelésre.
5. NF_REPEAT: a kapocs ismételt meghívása.

Ez a kód megmutatja, hogy mi történik a csomaggal, mielőtt a rendszerbe vagy onnan kiküldésre kerül.
Azt viszont még mindig nem tudjuk, hogy milyen információkat vagy beállításokat adhatunk a csomaghoz, és hogyan tehetjük azt meg. További kérdés, hogy vajon ezek a változtatások együttműködnek-e a jelenlegi megvalósítással.

IP-beállítások

Az internet-protokoll (IP) egy kevésbé ismert tulajdonsága, hogy egy IP-csomag változó hosszúságú (maximálisan 40 bájt) többletinformációt is tartalmazhat, amely a szabványos 20 bájtos fejrészt követi. Ez a kiegészítés az úgynevezett opcionális terület, amelynek egy része biztonsági információk tárolására van fenntartva.
Jelenleg az Internet Protokoll két biztonsági beállítást tartalmaz. Ezek közül az egyik a ***DoD Basic Security Option*** (alap biztonsági beállítás, 130-as beállítástípus), amely lehetővé teszi, hogy az IP-adatcsomagokat a titkosítás foka szerinti címkével lássuk el. Ez a beállítás 16 titkosítási fokozatot különböztet meg, amelyek különböző számú megszorítást tartalmaznak a csomag kezelésére vonatkozóan. További biztonsági információk kezelését, mint amilyen a biztonsági osztály és szakasz megkülönböztetése, a második biztonsági beállítás teszi lehetővé, amelyre ***DoD Extended Security Option*** (ESO, kiterjesztett biztonsági beállítás, 133-as beállítástípus) néven hivatkozik a szakirodalom. E két beállítás adatterületének értékeit az Információs Rendszerek Védelmi Irodája (***Defense Information Systems Agency***) hivatott nyilvántartani.
A számítógép forgalmazók ma már olyan kereskedelmi operációs rendszereket kínálnak, amelyek kötelező hozzáférés-ellenőrzéssel és többszintű biztonsági beállításokkal rendelkeznek, s ezek a rendszerek már nem kizárólag a védelmi vagy hírszerzési szervezetek számára készülnek, hanem olyan, nyilvánosan hozzáférhető kereskedelmi rendszereknek, amelyeket az állami és civil szféra is elterjedten használ.
A kisszámú ESO formátumkód nem támogatja a kereskedelmi biztonsági beállítások összes lehetséges alkalmazási módját. A BSO és ESO tervezésekor csak az Egyesült Államok Védelmi Minisztériumának támogatása volt a cél.
A különböző egyéb biztonsági módok támogatására a CIPSO (***Commercial IP Security Option***, kereskedelmi IP biztonsági beállítás) került kidolgozásra. Az internetes tervezet biztosítja a kötelező hozzáférés ellenőrzés (***mandatory access control***, MAC) biztonsági rendjéhez szükséges formátumot és eljárásokat.
A mi megvalósításunkban az adatcsomagok címkézésére használt IP-beállítások a FIPS 188 szabványon és a kereskedelmi IP biztonsági beállításon (CIPSO) alapulnak. Az általunk kifejlesztett rendszerben az IP fejrészt ezen szabványok szerint változtatjuk meg, hogy alkalmassá tegyük a biztonsági információk hozzáadására és hálózaton való küldésére.

A DSM IP-beállításai

Az IP-beállítások segítségével továbbítandó két biztonsági információ a biztonsági azonosító (***Security ID***, SID) és a biztonsági csomópont-azonosító (***Security Node ID***, NID). A DSM úgy módosít minden egyes IP-csomagot, hogy IP-beállításként ellátja azokat ezekkel a biztonsági információkkal. A *2. képen* látható ennek a módosított IP fejrésznek a felépítése.

A fejrész beállításainak listája a következő:

- CIPSO: 1 bájt, 134-es értékkel.
- Hossz: 1 bájt, amelynek értéke a teljes hossz, beleértve a típus és hossz mezőket is. A jelenlegi IP-fejrész 40 bájtos korlátja miatt ez az érték nem haladhatja meg a 40-et.
- Az értelmezési tartomány (DOI) azonosítója: előjel nélküli 32 bites egész. A 0 foglalt érték és nem jelenhet meg egy CIPSO beállítás DOI-azonosítójaként sem. A megvalósításoknak el kell fogadniuk, hogy a DOI-azonosítót nem köti semmilyen különleges bájtszintű korlátozás.
- A CIPSO értelmezési tartomány mező vagy a FIPS 188 szerinti biztonsági jelkészlet neve: ez 10001000 hexadecimális értékre van beállítva, ami egy önkényes érték, mivel jelenleg erre a területre nincs érvényes korlátozó szabály.
- Szabad formátum-jel: egy bájt, amely azt jelzi, hogy az ezt követő mezők olyan új mezők, amelyeket a szabvány nem határoz meg (vagyis tetszőlegesek). Értéke 7.
- Hossz: 1 bájt, a címkék teljes hosszát jelöli.
- Címkék (SID, NID): A CIPSO címkecsoportokat használ az IP-csomagban tárolt adatokra vonatkozó biztonsági információk tárolására. Minden címke egy címketípus azonosítóval kezdődik, a címke hosszával folytatódik, majd végül ezt követi a tényleges biztonsági információ.
- SID címke: címkeazonosító: 1 bájt (értéke 3), címkehossz: 1 bájt (értéke 6), címkeadat: a sid 32 bites értéke.
- NID címke: címkeazonosító: 1 bájt (értéke 6), címkehossz: 1 bájt (értéke 6), címkeadat: a nid 32 bites értéke.
- Az általunk használt IP-beállítás a CIPSO. Ezeket a mezőket a szabvány nem határozza meg, így használhatóak az általunk definiált módon.
- A DOI és a FIPS 188 szabvány szerint a következő mezők új mezők, amelyeket a szabvány nem határoz meg, ezért szabadon felhasználhatóak.

DSM hálózati kapcsok

A DSM-ben az LSM biztonsági kapcsokat használtuk fel az IP üzenetek biztonsági címkékkel történő ellátására. A következőkben ezt egy példán keresztül fogjuk szemléltetni, amelyben a program a hálózaton keresztül olyan csomagot küld, amelynek a foglalatát módosította. A program használja az eljáráskönyvtár néhány függvényhívását. Egy ponton egy rendszerhívás jön létre, amely az üzenetet a Linux rendszermagnak továbbítja. A rendszermag foglalat-megvalósításának belépési pontja a `sys_socketcall()` függvény, amely a ***net/socket.c*** fájlban található. A híváslánchan a ***net/socket.c*** fájlban lévő `sock_sendmsg()` függvény *(1. lista)* futtatása történik.

A függvény első tevékenységeinek egyike a biztonsági kapocs futtatása (***security_ops->socket_ops->sendmsg(...)***).
A kapocs a DSM foglalat-kapocsban ér véget, amely ai IP csomagot módosítja, mint az a 2. *listán* látható.
A `dsi_options_fill` függvény elindítja a biztonsági információt az átmeneti tár felé, ahogy azt az előző részben leírtuk. Az ezt követő függvények végrehajtása során kerül ez az információ az IP-üzenet beállítási részében rögzítésre. A SID a foglalat biztonsági azonosítójából származik, a NID pedig az egész csomópontban érvényes érték, vagyis nincs szükség arra, hogy a függvénynek paraméterként átadjuk.
Ezt követően a biztonsági információkat már tartalmazó módosított csomag a normál feldolgozó eljárásnak megfelelően kerül továbbításra a rendszermag felé, majd a hálózatra. A fogadó oldalon a bejövő üzenetek az `sk_buff` szerkezetnek megfelelően tárolódik és különböző függvények és kapcsok sorozata által kerül elő-feldolgozásra.
Az egyik ilyen függvény az `ip_options_compile` *(3. lista)* a ***/net/ipv4/ip_options.c*** fájlban, amely a beállításokat dolgozza fel.
A CIPSO esetén a decode_options biztonsági kapocs kerül meghívásra. Ez után következik a DSM `dsi_decode_options` kapocs, amelyben a bejövő csomag biztonsági paraméterei (SID, NID) kerülnek kiolvasásra és tárolódnak egy az sk_buff formátumhoz kapcsolódó biztonsági formátumban. A biztonsági információkkal feltöltött `sk_buff` tárolók a bejövő foglalatsorhoz csatlakoznak, és itt várjuk, hogy a fogadó program kiolvassa őket.
A kiolvasás érdekében a program egy `sys_socketcall ()` rendszerhívást bocsát ki, ugyanúgy, ahogy a küldött csomag esetén is tette. A hívás ismételten átmegy a DSM biztonsági kapcson, ahol a fogadó foglalat biztonsági azonosító érvényesítésre kerül a bejövő csomag `sk_buff` biztonsági tartalmával. Ha a foglalat a megadott biztonsági azonosítóval nem fogadhatja a csomagokat, akkor azok elvesznek.
A *4. listán* látható az *include/net/sock.h* fájlban lévő rendszermag-függvény.
Ebben láthatjuk, hogy a `sock_rcv_skb` biztonsági kapocs kerül meghívásra, amit a `dsi_sock_rcv_skb` DSM-függvény vált fel a DSM betöltődésekor. Ebben a függvényben történik a biztonsági érvényesítés. A példakódból láthatjuk, hogy milyen műveletekre van szükség a biztonsági címkék kezeléséhez.

Teljesítményvizsgálatok

Számos sebességtesztet végeztük arra vonatkozóan, hogy az IP-fejrész kibővítése a beállításainkkal befolyásolja-e az átfogó teljesítményt, és ha igen, milyen mértékben. Az egyik tesztben egy UDP-csomagot küldtünk át a telep csomópontjai között és mértük a csomag átviteléhez szükséges idő növekedését, ami a küldő oldalon a biztonsági beállítások módosításából, a fogadó oldalon pedig ezek kinyeréséből adódhatott.
A megvalósításunkon alapuló biztonsági információk hozzáadása okozta többletmunka átlagos értéke 30% volt. Ennek nagy részét (mintegy 25%-ot) az IP-csomag IP-beállításain végzett módosítása tette ki, a maradék többletmunka (körülbelül 5%) pedig a Linux rendszermag biztonsági kapocs-rendszeréből származik. Látható, hogy a többlet nagy rész az IP-csomag beállításainak a módosításából származik és csak kisebb rész írható a biztonsági kapcsok rendszerének számlájára.
A jövőbeli erőfeszítéseink arra fognak irányulni, hogy növeljük IP-módosító algoritmusunk hatékonyságát, miközben továbbra is az IP-beállításokat tekintjük a biztonsági információk átviteli eszközének.

Összegzés

Az IP-beállítások megváltoztatásával képesek voltunk arra, hogy a DSM segítségével biztonsági információkat továbbítsunk a telep csomópontjai felé. Az IP-csomagok módosításának optimalizálásával már első próbálkozásra is jelentős teljesítménynövekedést értünk el: a 30%-os teljesítménycsökkenést sikerült 14%-ra leszorítani. Ezek az eredmények igen ígéretesek, több olyan lehetőséget is látunk a további tökéletesítésre, amellyel még kisebb többletmunkával érhetjük el a célunkat. Mindenesetre az eredmények jól mutatják azokat a kihívásokat, amelyekkel egy hatékony osztott biztonsági rendszer fejlesztése során szembe kell néznünk. Reméljük, hogy minél többen próbálják majd ki az általunk kifejlesztett DSI és DSM technológiát és megosztják velünk a tapasztalataikat.

Köszönetnyilvánítás

Köszönjük David Gordonnak a Sherbrooke Egyetem munkatársának a DSM kifejlesztéséhez való hozzájárulását.

Linux Journal 2004. április, 120. szám

Ibrahim Haddad a Linux Journal társszerkesztője, a montreali Ericsson Research & Innovation Unit kutatója. Társszerzője a McGraw-Hill/Osborne által kiadott két Richard Peterson-könyvnek: Red Hat Linux Pocket Administrator és Red Hat Enterprise Linux & Fedora Edition: The Complete Reference (DVD edition).

Miroslaw Zakrzewski az új generációs CDMA rendszerek kifejlesztésén dolgozik az Ericssonnál a kanadai Montrealban. Elérhető a Miroslaw.Zakrzewski@Ericsson.ca címen.

KAPCSOLÓDÓ CÍMEK

A DSI és DSM honlapja:
➲ www.linux.ericsson.ca/dsi
A FIPS 188 szabvány:
➲ csrc.nist.gov/publications/fips/fips188.html
Cikkek a Linux csomagszűrőjéről:
➲ www.linuxjournal.com/article/4852 és
➲ www.linuxjournal.com/article/5617
Az LSM: ➲ lsm.immunix.org
Hálózati átmeneti tárak:
➲ www.linuxjournal.com/article/1312
Az Open System Lab oldala: ➲ www.linux.ericsson.ca
A SE Linux honlapja: ➲ www.nsa.gov/selinux