A Linux Capabilityk használata

A paranoiás rendszergazdák legnagyobb örömére ezúttal a felhasználók jogosultságainak korlátozásával, azon belül is a POSIX Capabilitykkel foglalkozunk.

Mostanában gyakori téma a biztonság, és erre minden okunk megvan. Minél jobban elterjednek a hálózatok és az internethasználat, a biztonság egyre fontosabbá válik. Mint minden jó rendszer, a Linux is folyamatosan fejlődik, hogy megfeleljen az egyre növekedő biztonsági elvárásoknak.
A biztonság egyik lépcsője a felhasználók jogainak megfelelő beállítása. A Unix-stílusú felhasználói jogosultságoknak két fajtája létezik: felhasználói és rendszergazdai, vagyis `root` jogosultság. A felhasználók alapértelmezetten semmilyen jogkörrel nem rendelkeznek: nem szólhatnak bele más felhasználók folyamatainak működésébe, és mások fájljait sem változtathatják meg. Az alkatrészek közvetlen elérése és a legtöbb hálózati lehetőség szintén korlátozott. Ezzel szemben a rendszergazda szinte bármit megtehet.
Előfordulhatnak azonban olyan esetek is, amikor köztes megoldásra lenne szükségünk. Megeshet, hogy egy folyamatnak feladata elvégzéséhez különleges jogosultságokra van szüksége, ugyanakkor a teljes körű rendszergazdai hozzáférés túlzás. A `ping` program például `setuid-root` jogosultsággal rendelkezik, így alkalmas ICMP-csomagok küldésére. A veszély abban rejlik, hogy még mielőtt a `ping` eldobná a rendszergazdai jogosultságokat, az esetlegesen benne lévő hibákat egy rossz szándékú program kihasználhatja, és ezáltal rendszergazdai jogosultságokat szerezhet.
Szerencsére most már létezik megfelelő köztes megoldás: a POSIX Capabilityk. Ezek a Capabilityk a rendszergazdai jogosultságokat logikus csoportokra osztják, melyeket tetszés szerint adhatunk ki a folyamatoknak vagy vonhatunk meg tőlük. A Capabilityk segítségével a rendszergazda pontosan meghatározhatja, hogy egy folyamat milyen műveleteket hajthat végre, és ily módon jelentős mértékben csökkenthető a rendszert fenyegető kockázat. Ha szerencséd van, foltozásra sem lesz szükséged.
Az összes Capability listája a */usr/include/linux/capability.h* fájlban található meg, ahol a felsorolás a `CAP_CHOWN`-nal kezdődik. Az egyes sorok jelentése elég nyilvánvaló, ráadásul minden eléggé jól le van írva. Az egyes Capabilityk csak egyszerűen bitek egy bittérképen. Ez a bittérkép jelenleg 32 bit hosszú, és 28 bit már használatban áll. Jelenleg éppen arról folynak viták, hogyan bővítsék e bittérképet.

A Proc felület

Jelenleg, a 2.4.17-es rendszermag idejében a */proc/sys/kernel/cap-bound* állomány egy 32 bites integert tartalmaz, ami a pillanatnyi rendszerszintű Capability-készletet jelöli. Ez a globális Capability-készlet határozza meg, hogy a rendszeren futó programok milyen képességeket birtokolhatnak. Ha egy Capabilityt lecsippentünk, a rendszeren lévő összes folyamat képtelen lesz az ehhez a Capabilityhez kapcsolódó műveletek végrehajtására, a rendszergazdai folyamatokat is beleértve! Ha például valaki betör a rendszerünkbe, sok esetben első dolga, hogy egy úgynevezett támadócsomagot (rootkit) telepít, vagyis egy olyan eszközkészletet, amellyel elrejtheti a kiszolgálón végzett tevékenységét és tovább fertőzhet, illetve hátsó kaput építhet a rendszerbe – ezen keresztül később bármikor visszatérhet. Ezt követően olyan modult tölt be a rendszermagba, amellyel saját folyamatait leplezheti. A rendszergazda, hogy ezt megakadályozza, a rendszerindítási folyamat utolsó lépéseként a `CAP_SYS_MODULE` Capabilityt eltávolíthatja a rendszerből. Ezzel a lépéssel újabb modulok betöltése vagy eltávolítása hiúsítható meg. Ha egyszer egy Capabilityt eltávolítottunk, nincs mód az újbóli hozzáadására. Amennyiben összes képességét mégis vissza szeretnénk állítani, a rendszert újra kell indítani (ezt a `CAP_SYS_BOOT` Capability eltávolításával ugyancsak meggátolhatjuk, ezt követően a rendszert már csak magán a gépen lehet kikapcsolni).
Rendben, lódítottam. Két mód is létezik arra, hogy visszaállítsunk egy Capabilityt:

1. Az `init` elvileg képes Capabilityk újbóli felvételére – legjobb tudomásom szerint azonban ennek megvalósítása még várat magára. Erre a tulajdonságra Capabilityk kezelésére felkészült rendszerek esetén lehet szükség, amennyiben meg akarjuk változtatni a futási szintet.
2. Ha egy folyamat rendelkezik a `CAP_SYS_RAWIO` Capabilityvel, akkor a */dev/mem* állományon keresztül képes módosítani a rendszermag memóriáját, s így olyan jogosultságokat biztosíthat magának, amilyeneket csak akar. Természetesen ezt a Capabilityt is eltávolíthatjuk, de vigyázzunk, mert így egyes programok (pl.: az X) nagy valószínűséggel működésképtelen lesznek.

A `cap-bound` kézi szerkesztése elég fárasztó, szerencsére azonban létezik egy lcap nevű eszköz, mely a Capabilityk beállítását hivatott megkönnyíteni. Lássunk egy példát a `CAP_SYS_CHOWN` eltávolítására:

```
lcap CAP_SYS_CHOWN
```

Ezután már x egy fájl tulajdonosát képtelenség megváltoztatni:

```
chown nobody test.txt
chown: changing ownership of  test.txt :
➥Operation not permitted
```

A három felsorolt kivételével az összes Capability ilyen módon távolítható el:

```
lcap -z CAP_SYS_BOOT CAP_SYS_KILL
➥CAP_SYS_NICE
```

Fontos, hogy a `cap-bound` módosításával csak az újonnan induló programok jogait korlátozhatjuk, pontosabban csak azokét, amelyek az `exec(2)`-t meghívják (vess egy pillantást a rendszermag forrásában található *fs/exec.c* fájlban a `compute_creeds` függvényre).
Azok a programok, amelyek már korábban is futottak, jogaikat megtartják.
A következő részben a létező folyamatok képességeinek megváltoztatásáról és a már említett csapdáról szólunk. Amennyiben az `lcap` parancsot kapcsolók nélkül futtatjuk, kiírja, hogy jelenlegi rendszerünk milyen képességekkel rendelkezik. Ha a beállításaidban a `CAP_SETPCAP` nincs bekapcsolva, akkor rendszermagodon először végre kell hajtanod

egy apró változtatást. A rendszermag forrásában az ***include/linux/capability.h*** állományban cseréld le a következő két sort (lásd a ➲ http://www.linuxvilag.hu/capability webhelyen). Ezt követően rendszermagodat fordítsd újra.

Valójában annak is megvan az oka, hogy `CAP_SETPCAP` alapértelmezetten ne legyen engedélyezve: éles rendszeren egy beállítás biztonsági kockázatot jelenthet. Bár már létezik folt ennek önműködő végrehajtására, e cikk írása idején hivatalosan még nem került be a rendszermagba. A biztonság kedvéért, miután a próbálgatást befejezted, ezt a Capabilityt távolítsd el a rendszeredből!

A cikk írása idején a folyamatok Capabilityinek megváltoztatására két utasítás áll a rendelkezésünkre: a `capset` és `capget` rendszerhívások. Ez a későbbiek folyamán még változhat. Ha szeretnénk, hogy alkalmazásaink más rendszereken is működjenek, a libcap használata ajánlott (➲ http://www.kernel.org/pub/linux/libs/security/linux-privs/kernel-2.4)

A `capset` prototípusa:

```
int capset(cap_user_header_t header, const
➥cap_user_data_t data);
```

A `header` kapcsoló nagyon ötletes módja a folyamat kijelölésének:

```
typedef struct __user_cap_header_struct {
        __u32 version;
        int pid;
} *cap_user_header_t;*
```

Ha a `pid` értéke -1, akkor az összes jelenleg is futó folyamat módosul. Ha az érték ennél kisebb, akkor az a folyamatcsoport változik, amelynek azonosítója `pid * -1`. Az alapötlet tehát ugyanaz, mint a `kill(2)` esetén.

A `data` kapcsolóval jelölhető ki, hogy melyik Capability-készleten szeretnénk módosítani. Három közül választhatunk (lásd a ➲ http://www.linuxvilag.hu/capability webhelyen).

A `permitted` azokat a Capabilityket jelöli, amelyről a programnak tudomása van.

Az `effective`-készlet azokat a Capabilityket határozza meg, amiket a folyamat a `permitted`-készletből tényleges felhasználhat. Olyan ez, mintha egész seregnyi költővel rendelkeznél (ez a `permitted-kØszlet`), de csak néhányat választhatsz ki közülük, melyekkel védheted magad (mondjuk Alan Ginsberget, ez az `effective`-készlet).

Az `inheritable`-készlet azokat a Capabilityket jelölik, amelyeket tovább örökíthetünk az `exec(2)`-kel indított folyamatokra. A `fork(2)` függvény nem változtat a Capabilityken, a létrejövő gyermekfolyamat pontosan ugyanazokkal a képességekkel rendelkezik, mint a szülője.

Csak azokat a Capabilityket vehetjük fel egy folyamat `effective`- és `inheritable`-készletébe, melyeket a `permitted`-készlet is tartalmaz. A `permitted`-készlet csak akkor módosítható, ha a folyamat rendelkezik a `CAP_SETPCAP` Capabilityvel.

Sajnálatos módon a Capabilityk egyelőre semmilyen fájlrendszert nem támogatnak, így használatuk ezen a területen nagyon korlátozott. De eljön majd az idő, amikor a rendszermagok képesek lesznek rá, hogy a program fájlleírójában (inode) tárolják az ahhoz tartozó Capabilityket, így véve elejét a `setuid` biteknek, amelyeket oly sok rendszerprogram igényel.

Ha ez működni fog, a `ping` parancsot ilyen egyszerűen lehet majd feljogosítani arra, hogy alacsony szintű foglalatokat használhasson:

```
chattr +CAP_NET_RAW /bin/ping
```

Sokkal sürgetőbb gondok miatt ennek kidolgozására sajnos még nem jutott idő.

Ha kedved van hozzá, a libcapet kedvenc szolgáltatásaid jogainak megnyirbálására is használhatod, és csak azokat a Capabilityket kapják meg, amikre tényleg szükségük van. Az `xntpd`-hez jó néhány ilyen folt létezik; néhány ekképpen módosított változathoz még külön rpm-csomag is létezik. A Google-lal rákereshetsz kedvenc programjaid Capabilitykre felkészült változataira, amelyekről úgy érzed, hogy a legnagyobb lyukat jelentik rendszered biztonságán.

A `setpcap` rendszerhívással egy már futó folyamat képességeit változtathatod meg. Ha például egy általános héj (shell) PID-je 4235, akkor ily módon ruházhatod fel azzal a képességgel, hogy minden folyamatnak képes legyen jelzést küldeni:

```
setpcaps ·cap_kill=ep· 4235
```

Tételezzük fel, hogy egy barátunk kipróbált egy CGI-programot, és az őt futtató Apache állandóan végtelen ciklusba keveredik – ekkor biztosíthatjuk számára, hogy a megbolondult Apache-okat kilője. Ezt elegendő egyszer beállítanod a bejelentkezési héjra, utána el is feledkezhetsz róla.

Most pedig azt szemléltetjük, hogy az `execcap` és `sucap` felhasználásával hogyan futtathatjuk a `ping`-et `nobody`-ként, egyedül a `CAP_NET_RAW` képességet engedélyezve számára. Pingünk célpontjául a válasszuk ➲ http://www.yahoo.com-ot:

```
execcap ·cap_net_raw=ep· /sbin/sucap nobody
➥nobody /bin/ping www.yahoo.com
```

Ez a példa nem különösebben hasznos, lévén futtatásához rendszergazdai jogosultságokkal kellene rendelkezned, de nagyszerűen megmutatja a lehetőségeidet. Ezektől a hátrányoktól eltekintve ezt az eszközt a rendszergazdák rendszerük biztonságának növelésére nagyon jól használhatják. Például a `CAP_SYS_BOOT`, `CAP_SYS_RAWIO` és a `CAP_SYS_MODULE` nélkül futó rendszer rendkívül megnehezíti a támadó dolgát. Nem nyúlhatnak bele a rendszermag memóriájába, nem indíthatnak új modulokat, de még a rendszert sem indíthatják újra, hogy egy hátsó kapukkal megtűzdelt rendszermagot betöltsenek.

Ha rendszered naplófájljai `'append-only'`-ra vannak beállítva (vagyis csak újabb sorokat lehet hozzájuk fűzni), rendszerprogramjaid pedig `'immutable'`-re (vagyis megváltoztathatatlanra), ráadásul még a `CAP_LINUX_IMMUTABLE` Capabilityt is eltávolítottad, a behatoló képtelen lesz eltüntetni a nyomait, illetve a saját módosított rendszerprogramjait sem tudja feltelepíteni. A forgalomelemző eszközök – mint amilyen a `tcpdump` is – használhatatlanná válnak, ha a `CAP_NET_RAW`-képességet eltávolítjuk. Ezenkívül a `CAP_SYS_PTRACE`-t is tüntesd el, ezzel programjaid nyomkövetését is letiltod. Az ilyen barátságtalan rendszerek a kódtörő kölykök (script kiddie) rémálmát jelentik, egyetlen lehetőségük, hogy lekapcsolódnak a rendszerről, és megvárják, amíg felfedezik őket.

Összegzés

Ezekkel a képességekkel a nagyon bonyolult szolgáltatásokat is finomhangolhatjuk, így rendszerünk biztonságát minden szempontra kiterjedően testreszabhatjuk. Ha más nem is, de a paranoiás rendszergazdák olyan eszközhöz jutnak, mely megkönnyíti a harcot az „ellenük” vívott véget nem érő küzdelemben.

Linux Journal május, 97. szám

Michael Bacarella
(mike@bacarella.com) a Netgraft cég elnöke, mely webes rendszerek fejlesztésével és biztonsági elemzéssel foglalkozik. New Yorkban él csodálatos menyasszonyával és a Kang nevű, legrémisztőbb iguánával.