FreeBSD – a szomszéd vár (10. rész)
A vár őrei

Egy kiszolgáló gép elsődleges feladata, hogy a kliensek kéréseit kiszolgálja, s eközben meg kell különböztetnie a klienseket a támadóktól. A BSD vár önmagában elegendő bizonyos fokú védelemre, azonban ezt érdemes kibővíteni.

Betörési pontok

Több támadási típusra kell felkészülnünk, amelyek gyakran szolgálnak betörési útként különféle támadóknak, legyenek humán betörők, vagy jól megírt programok (férgek vagy vírusok).

A ***szolgáltatásmegtagadás (Denial of Services, DoS)*** célja, hogy elfogyassza a gépünk egyes erőforrásait, amelyek fontosak az üzemszerű működéshez. Többnyire nyers erőt felhasználva annyi kérést zúdítanak a célzott gépre, hogy az képtelen lesz a kiszolgálásra, esetleg össze is omlik. Kisebb arányt képviselnek azok a támadások, amelyek egy szolgáltatás programozási hibáját kihasználva egyetlen hálózati kérést küldenek csupán, s ez már elegendő arra, hogy akár a kiszolgáló is a padlóra kerüljön.

Ez utóbbi esetben az adott szolgáltatás vagy a rendszermag hibáját kijavítva a ***DoS*** lehetősége megszűnik. Az előbbi esetben érdemes korlátozni az egyes szolgáltatások által használható erőforrásokat, így a ***DoS*** csak ideiglenesen jár sikerrel.

A felhasználói jog szerzése sokkal gyakoribb és elterjedtebb támadási mód. Sok olyan szolgáltatás fut még, amely a felhasználói adatokat (név és jelszó) titkosítás nélkül küldi és fogadja (***FTP***, ***POP3***, stb). Ezeket az információkat a hálózati forgalom figyelésével könnyedén le lehet jegyezni, s később felhasználni. Egy bizonyos felhasználói számot túllépve a rendszergazda már nem lesz képes kideríteni, hogy valóban az adott felhasználó lépett be egy késő esti órában dolgozgatni, vagy sikeresen ellopták az adatait.

Előfordulhat, hogy a root felhasználó nevében sikerül belépnie a támadónak, akár egy hibás program biztonsági hiányosságát kihasználva, akár a jelszó figyelésével. Számolnunk kell azzal, hogy a rendszergazdai jogok birtokában könnyedén eltávolíthatja a betörés és az adatlopás nyomait. Ritka eset, amikor a támadók összeomlásra késztetik a kiszolgálót magát, gyakoribb, hogy hátsó ajtót hoznak létre, s azon már észrevétlenül képesek közlekedni. A hátsó kapuk gyakorta más gépek feltöréséhez adnak rejtőzési lehetőséget a profi adatgyűjtők számára.

Alapvető biztonság

Ha nem tesszük biztonságossá a root hozzáférést, akkor felesleges dolgoznunk a többi alrendszer biztonságossá tételén, a támadóknak tálcán kínáljuk rendszerünk leginkább áhított gyenge pontját. A legtöbb kiszolgáló esetén nem fontos a root hozzáférést konzolon kívül hozzáférhetővé tenni, ezen túllépve még akár a konzolon is megtilthatjuk, s így csak egy meghatározott felhasználón át tudunk rendszergazdai jogokhoz jutni, mégpedig a su parancsot használva. A távoli hozzáférést a /etc/ttys állományban tudjuk korlátozni, így az általános programokat tekintve (***telnet***, ***rlogin***, stb.) már kizártuk azon próbálkozásokat, amelyek egyenesen a root felhasználó jogaival szeretnének bejutni. Egyéb esetben a programok és szolgáltatások saját beállításaival tudunk még sikereket elérni: például az ***SSH*** esetén a ***/etc/ssh/sshd_config*** állományban a `PermitRootLogin` változó értékét állítsuk **NO** állapotra.

Ha sikeresen megtiltottuk a közvetlen rendszergazdai belépéseket, akkor sajnos eléggé jól elvágtuk magunkat a szervertől hálózaton át: ugyanis akárki nem lehet a `su` parancs segítségével rendszergazdai jogok birtokosa:

```
tomy66@szerver ~ $ su -
su: Sorry
```

A `su` parancs ugyanis azonnal visszautasítja a root jogok felé irányuló kéréseket, még a jelszót sem kéri el. A ***FreeBSD*** külön csoportban kezeli azon felhasználókat, akik képesek rendszergazdai jogokat szerezni: a `wheel` csoport szolgál erre a célra:

```
tomy66@szerver ~ $ grep wheel /etc/group
wheel:*:0:root,auth.gabor,franko
```

Ha az aktuális felhasználó nincs a megadott csoportban, akkor sajnos (illetve szerencsére) nem lehet

belőle rendszergazda az adott gépen. A csoportba tartozó felhasználókat viszont jelszó ellenében már közel engedi a tűzhöz:

```
auth.gabor@szerver ~ $ su -
Password:
```

Ezzel a módszerrel az adott felhasználó(k) képesek lesznek belépni a kiszolgáló gépre, s a saját jelszavukat megadva, majd a root jelszót megadva már rendszergazdai joggal bírnak. Sajnos azonban mind a két jelszót be kell gépelni a billentyűzeten, így az kifigyelhető és felhasználható betörési célokra. Ennek okán érdemes a ***SSH*** esetén az adott felhasználó jelszavát megszüntetnünk (nem üresre állítanunk!). Ennek módja egyszerű, a `vipw` parancs segítségével szerkesztjük a ***passwd*** állományt (gyakorlatilag a /etc/master.passwd állományt látjuk ekkor), majd a megfelelő felhasználónévnél az alábbi sort

```
auth.gabor:$1$tLlpY8GZ$HOPQAc3eqNy1adsR1XmqB1:1001:
➥1001::0:0:Auth
Gábor:/home/tanar/auth.gabor:/usr/local/bin/bash
```

így módosítjuk:

```
auth.gabor:*:1001:1001::0:0:Auth
➥Gábor:/home/tanar/auth.gabor:/usr/local/bin/bash
```

A beírt és kódolt jelszó soha nem lesz azonos a * karakterrel, így tetszőleges jelszóval sem lehet majd belépni ezzel a felhasználóval. Kell készítenünk egy kulcspárt (egy privát és egy publikus kulcsot), mégpedig az `ssh-keygen` parancs segítségével. A publikus kulcsot hozzá kell másolnunk a kiszolgáló gép felhasználójának ***.ssh*** könyvtárában lévő ***authorized_keys*** állományához. A kliens gép ***.ssh*** könyvtárába pedig a privát kulcsot kell őriznünk, ***identity*** néven.
Ha be szeretnénk lépni ***SSH***-n keresztül a kiszolgáló gépre, akkor az ***SSH*** nem lesz képes jelszavas azonosításra, hiszen bármilyen jelszót is írnánk be, az nem lesz megfelelő. Ellenben a kulcspáron alapuló hitelesítés működik, így ha az ***identity*** állományban lévő privát kulcsból előállított publikus kulcs megtalálható a kiszolgáló ***authorized_keys*** állományában, akkor jelszó és kérdés nélkül be fog engedni. A feladatunk mindössze annyi, hogy ne engedjük ki a kezünkből a privát kulcsot:

```
$ ssh auth.gabor@szerver
Last login: Thu Jun 30 09:31:28 2005 from kliens
szerver:~ $
```

Mindig csak annyi szolgáltatást futtassunk, amennyi szükséges; s mindig figyeljünk oda az újabb verziókra, hibajavításokra. Egy régen frissített szolgáltatás akár egy széles és egyenes út lehet a támadók számára, hiszen lehetnek benne biztonsági hiányosságok. Figyeljük rendszeresen a biztonsági értesítőket, hiszen órákon is múlhat rendszerünk épsége. Ha megállapítottuk, melyek a szükséges szolgáltatások, akkor lehetőleg mindegyiket tegyük egy ***jail*** által létrehozott „homokozóba" (***sandbox***). Ezek a „homokozók" nem jelentenek túl nagy biztonságot, azonban a támadók először ezekbe kerülnek, és innen is tovább kell jutniuk a rendszerünk belső részei felé. Törekedjünk minél több ilyen réteg elkészítésére, így egyre több időt kell a támadóknak a rendszer megismerésével és feltörésével eltöltenie, így a mi rendszerünk egyre kevésbé lesz vonzó számukra.
A ***FreeBSD*** operációs rendszer sok szolgáltatást alapból homokozóban futtat (***ntalkd***, ***fingerd***, stb.), így ezekkel nem is kell törődnünk. További hibaforrás a ***SUID/SGID*** bitekkel ellátott programok telepítése, mivel az ezekben előforduló hibák automatikusan rendszergazdai jogokat biztosítanak a felhasználók számára. Jól jegyezzük fel, hogy ezen programok közül mennyit telepítünk fel, illetve ezek hol találhatók: ha a méretük megváltozik, akkor annak komoly következményei lehetnek.

Kiterjesztett biztonság

Amint az alaprendszert biztonságossá varázsoltuk, még közel sem végeztünk a munkával. A ***FreeBSD*** programtelepítésének alapja a ports adatbázis, amely már 12.000 programcsomag felett tart. Éles rendszer esetén több problémával is szemberülhetünk a ports használatával, amelyek közül az egyik nagyon lényeges: nem értesülünk az újabb csomag kibocsátásának okáról. Egy egyszerű ***portupgrade*** végrehajtásakor az összes változott csomagot frissíthetjük, ám a ***„Ha működik, ne javítsd!"*** aranyszabályt ezzel csúfosan megszegjük. Többnyire nincs probléma, mivel az újabb programcsomag kompatibilis a régebbivel, ha azonban akadtak kisebb változtatások, akkor hamar és sürgősen kellett a hibát kijavítani (vagy visszatenni a régebbi verziót).
Ezen a problémán tud segíteni a ***security/portaudit*** csomag, amely a ***periodic*** programon keresztül rendszeresen elvégzi a telepített csomagok összevetését a rendelkezésre álló csomagokkal, s jelzi a csere szükségességi fokát.
A program telepítése után azonnal kérhetünk egy jelentést a frissítésre váró programokról, egyszerűen a

```
$ /usr/local/sbin/portaudit -Fda
```

parancsot kell kiadnunk. Ezek után a program egy új adatbázis letöltésével kezdi a munkáját, amely a frissítésre váró programok adatait tartalmazza:

```
auditfile.tbz           100% of   26 kB   27 kBps
New database installed.
Database created: 2005 Jún 30 Csü 18:40:13 CEST
```

Ezt követi azon csomagok felsorolása, amelyek esetén biztonsági okból javasolt a frissítés:

```
Affected package: ruby-1.6.8.2004.07.28_1
Type of problem: ruby -- arbitrary command
➥execution on XMLRPC server.
Reference: <http://www.FreeBSD.org/ports/
➥portaudit/594eb447-e398-11d9-a8bd
➥-000cf18bbe54.html>
```

A felsorolt programok kapcsán kettő lehetőségünk adódik: frissítjük vagy eltávolítjuk őket. Egyéb esetben a rendszerünk biztonsági oldalról hibás és hiányos marad, s nyitva

hagyjuk a kaput a támadók számára. Érdemes a program által kiemelt ***URL*** meglátogatása, s így bővebb információkat is tudunk szerezni a biztonsági fenyegetettségről. Esetleg létezik kerülő megoldás is (***workaround***), amely egy kissé kitolhatja a kockázatos frissítés idejét, de ez többnyire a szolgáltatás minőségromlásával jár.
Mivel a program minden nap lefut legalább egyszer, a fenti műveletekről levelet kapunk (ha olvassuk a root felhasználónak írandó leveleket :), és így a *„security run output”* tárgyú levélben olvashatunk a telepített csomagok biztonságosságáról.

BSD Process Accounting

Sok felhasználó esetén célszerű a futtatott programok nyomkövetése, így több napra, hétre visszamenőleg képesek leszünk lekérdezni, hogy melyik felhasználó – melyik programot – mennyi ideig használta.
A *Process Accounting* bekapcsolásához egyszerűen a */etc/rc.conf* állományba bele kell tennünk a

```
accounting_enable="YES"
```

sort, s ezzel a következő indításkor már futni is fog a szolgáltatás. Az azonnali bekapcsolásához az

```
$ accton
```

parancsot tudjuk használni. Ettől az időponttól kezdődően nyomon tudjuk követni felhasználóink tevékenységét:

```
$ lastcomm auth.gabor
[...]
kwebdesktop auth.gabor 2.22 secs Thu Jun 30 18:43
sh          auth.gabor 0.00 secs Thu Jun 30 18:41
[...]
```

Érdemes a *ports* adatbázis *security* kategóriáját figyelmesen átnézni, és a szükséges programokat használni, így ezzel is csökkentjük a biztonsági fenyegetettségeinket.

Auth Gábor (auth.gabor@enaplo.hu)
Egy pécsi középiskolában informatikát és programozást oktat. Tíz éve botlott először a UNIX rendszerekbe, 7 év Linux használat után kapta el a FreeBSD lázat, amiből máig nem tudott kigyógyulni.

KAPCSOLÓDÓ CÍMEK

A FreeBSD projekt honlapja: ➲ http://www.freebsd.org

A magyar FreeBSD honlap: ➲ http://www.freebsd.hu

A magyar BSD honlap: ➲ http://www.bsd.hu

A kézikönyv magyar fordítása
➲ http://www.enaplo.hu/FreeBSD/handbook/