Démonok harca – Gyakorlati útmutató az OpenSSH beállításainak megerősítéséhez

Elegendő néhány egyszerű trükk, és nyugodtan hajthatjuk álomra a fejünket: nem kell félnünk attól, ki tör be az éjjel féltve őrzött SSH kiszolgálónkra.

Ha egy *Linux* gépet kell felügyelnünk, akár hivatalból, akár mert a miénk, jó esély van arra, hogy olykor-olykor távolról be kell jelentkeznünk, rosszabb esetben folyamatos kapcsolatra lehet szükség. Legyen szó otthoni munkaállomásról, munkahelyi kiszolgálóról vagy hobbigépről, majdnem biztos, hogy a *Linuxhoz* való távoli hozzáférés esetén a távoli gépen *OpenSSH* kiszolgáló, a helyi gépen pedig valamilyen *SSH* ügyfél fut. (Ha mégsem, akkor pedig különösen ajánlom a cikk elolvasását.) Igaz ugyan, hogy az *SSH* kiszolgálók és ügyfelek hihetetlen mennyiségű forgalom titkosításáról gondoskodnak, de az is tény, hogy egyben minden kiszolgáló ajtó is, melyet a rossz fiúk szeretnének kinyitni. Közülük a legrosszabbak azok a démonnak nevezett egyének vagy csoportok, akik rossz szándékkal keresik a nevükben azonos kiszolgáló programokat.
Ez utóbbi démonok, a beállításoktól függően, vagy biztonságosak, vagy nem. Míg a hackerek általában szakmai büszkeségétől vezérelve törnek be gépekre, a crackerek infohuligánok, akik ugyanezt aljas szándékkal követik el. Így vagy úgy, egyik sem kívánatos jelenség. A továbbiakban az *OpenSSH* alapbeállításából kiindulva bemutatom, hogyan kell azt módosítani a számítógépünk védőbástyáját jelentő *OpenSSH* démon biztonságossá tételéhez.
A mi gépünkön lévő biztonsági rés ráadásul mások számítógépeire is veszélyt jelenthet. Az Internet a bolygónkon található adattovábbító megoldások egyik legveszélyesebbike.
Az *OpenSSH* segítségével megoldható a biztonságos kommunikáció a nem biztonságos csatornán. A beállítások főszereplője az *sshd_config* fájl, amely számos kapcsolót tartalmaz a biztonság fokozásához. Azt gondolhatnánk, hogy egy távoli hozzáférést biztosító eszköz eleve biztonságos, de ez sajnos távolról sincs így. Az *OpenSSH* kiszolgáló telepítését követően egy olyan, viszonylag biztonságos beállításunk lesz, amit a rendszergazda azért még jelentősen javíthat.
A távoli hozzáférés megtervezésekor a következő fő szempontokat kell figyelembe venni:

1. Kinek engedélyezzük a szolgáltatást?
2. Hogyan biztosítjuk a hozzáférést?
3. Honnan lehet igénybe venni?

A következőkben az *OpenSSH* kiszolgálót használjuk távoli hozzáférésre, így két kérdés marad megválaszolatlanul: kinek engedélyezzük a hozzáférést és honnét? A válasz lehet nagyon egyszerű, például egyetlen felhasználónk van egy adott tartományban. Lehet azonban meglehetősen komplikált is, ha például több, gyakran utazó felhasználónk van.

Az ördögi parti résztvevőlistája

Közismert, hogy egy *Linux* gép első és legfontosabb felhasználója a root, és remélhetőleg az is közismert, hogy ha root jogosultságokra van szükségünk, annak számos jobb módja van, mint *SSH*-val szimplán rootként bejelentkezni. Elég a nyers erővel elkövetett betörési kísérletre gondolnunk, melynek a root a legnyilvánvalóbb célpontja. A felhasználói név adott, azon már nem is kell gondolkodni. Az *sshd_config* fájlban a `PermitRootLogin` direktívával megtilthatjuk a rootnak a bejelentkezést.
A távoli hozzáférésre használt felhasználóknál nagyon ül a régi mondás: ***„A legjobb gyógymód a megelőzés.”***
Az *sshd_config* fájl `UsersAllow` és `UsersDeny` kapcsolóinak alkalmazása már több mint megelőzés, és bár ez minden egyes felhasználó felvételénél plusz egy lépést jelent, a `UsersAllow` módosítása azt jelenti, hogy a gyógymód ismeretére valószínűleg és remélhetőleg sohasem lesz szükség.
A `UsersAllow` direktívába nem csak csupasz felhasználóneveket tehetünk, hanem gépnévhez kapcsoltakat is. Tehát ha előre tudjuk ki és honnan szeretne a gépünkre bejelentkezni, a nyugalmunk érdekében minimális időráfordítással eszközölhetjük ezeket a restrikciókat az *sshd_config* fájlban.
A létező felhasználók listája az */etc/passwd* fájlban, illetve *NIS* vagy *LDAP* környezetben az ezzel egyenértékű állományban található. Az *1. Lista* bemutatja, hogy az elmúlt hónap biztonsági naplófájljaiban hogyan kereshetjük meg azokat a felhasználókat, akik sikeresen jelentkeztek be *SSH*-val.

Tartsuk szemmel és távol a démonokat

A démonokat kereső démonok figyelése viszonylag egyszerű, és a kapcsolódó alapbeállításokat mindenképp ajánlott átvezetni az *sshd_config* fájlba. Ha a `SyslogFacility` értéke `AUTH`, akkor az *sshd* démon a rendszernaplózás segítségével bejegyzéseket készít

1. Lista Sikeres bejelentkezések keresése a naplókban

```
cat secure* | grep Accepted | awk -F' ' '\
{print $1" "$2" "$9}' | uniq -u
Aug 30 juser
Aug 22 kuser
Aug 23 user
Aug 15 foo
...
Aug 24 13:23:19 foohost sshd[16348]: Accepted
password for phil from 127.0.0.1 port 47338 ssh2
Aug 24 13:23:25 foohost sshd[16398]: User root
not allowed because not listed in AllowUsers
```

2. Lista Az sshd_config fájlban nem engedélyezett felhasználók belépési kísérleteinél keletkezett naplóbejegyzések

```
User mail not allowed because not listed in
AllowUsers
User adm not allowed because not listed in
AllowUsers
```

3. Lista Nem létező felhasználók belépési kísérletéből származó naplóbejegyzések

```
Aug 28 06:04:15 foo sshd[11602]: Failed password
for illegal user a... from 10.0.0.1 port 35078 ssh2
Aug 28 06:04:17 foo sshd[11604]: Illegal user aaa
from 10.0.0.1
Aug 28 06:04:19 foo sshd[11604]: Failed password
for illegal user aaa from 10.0.0.1 port 35417 ssh2
Aug 28 06:04:21 foo sshd[11606]: Illegal user qqq
from 10.0.0.1
```

a */var/log/messages* és a */var/log/secure* fájlokba (az útvonalak és a fájlok neve függhet a disztribúciótól). Melegen ajánlom, hogy a rendszernaplót a ***Psionic*** naplóböngészőjéhez hasonló eszközzel nézzük meg, ez ugyanis értelmezi a naplóállományt, így képesek leszünk felismerni az ***sshd*** által engedélyezett és a sikertelen bejelentkezéseket egyaránt. Nyilvánvaló különbség van az engedély nélküli és a sikertelen belépés között. Utóbbi azt jelenti, hogy a gépen létező felhasználói fiók tulajdonosa sikertelenül próbálkozott, és az előző pont ilyen lehet. Ez meglehetősen fontos lehet akkor, amikor felhasználóneveket választunk, és azon elmélkedünk, ki kerüljön az `AllowUser`, és ki a `DenyUser` listára. Például lehet, hogy nincs `amanda` felhasználónk, de használjuk az `amanda` nevű nyílt forráskódú archiválóprogramot. Ha ez a név nem kerül a `DenyUser` listára, akkor a sikertelen bejelentkezések számát fogja növelni, nem pedig a nem létező felhasználókét. Biztos, ami biztos, a rendszer felhasználói fiókjait mindig a `DenyUsers` listára teszem, még akkor is, ha ez nem kifejezetten szükséges. A *2.* és a *3. Listában* az ***sshd*** naplójának a legfrissebb bejegyzései arról árulkodnak, hogy a rendszer felhasználói fiókjaival próbálkoztak.

Ha már úgyis a felhasználóneveknél tartunk, érdemes megemlékezni annak a lehetőségéről, hogy a gonosztevő démonok a belépéshez szükséges két dolog egyikével rendelkeznek. Egy felhasználóknak helyet adó webkiszolgáló esetében vagy az elektronikus levelek fejlécéből viszonylag egyszerűen hozzájuthatunk egy bizonyos géphez tartozó felhasználónevekhez. A démon ezzel a felhasználónév-jelszó páros egyik felének máris a birtokában van, így a támadás túllépheti a jól ismert felhasználói fiókok feltörésén, és következhetnek a nyers erővel folytatott kísérletek azokkal a felhasználónevekkel, amelyekről kiderült, hogy az adott számítógépen valóban léteznek.
Nem kell újra felfedezni a protokollt, elegendő a beállítások finomítása. Az ***OpenSSH*** erős titkosítást biztosít a hálózatba kötött végpontok között. A fel-felbukkanó biztonsági réseket folyamatosan javítják és elérhetővé teszik. Az egyik legfontosabb kiadás az ***SSH v1***-ről az ***SSH v2***-re váltás volt. A v1-es és a v2-es sshd démonnak is saját kulcsa van, más szóval, az ***SSH1*** és az ***SSH2*** nyilvános és titkos kulcsai nem keverednek, egymástól teljesen függetlenek. Ahogy a kulcsok, úgy a protokollverziók is függetlenek egymástól, és a konfiguráció `Protocol` direktívájával engedélyezhető az egyik, a másik, vagy mindkettő. Az ***SSH1*** már leáldozóban van, de még még mindig sok szervezetnél és alkalmazásban használatos. Érdemes ezt ellenőrizni az ***sshd_config*** fájlban, és letiltani az v1 verziót, amennyiben nincs rá szükség. Így nyilvánvalóan kiküszöbölhető, hogy a v1-es változat egy biztonsági hibájának áldozatául essünk.
A ***passwd*** fájl korábbi átvizsgálásánál talán szemet szúrt az ***sshd*** felhasználó, */var/empty* saját könyvtárral. Ha nincs, akkor a következő sorok különleges fontosak lesznek, és az Olvasó bizonyára további részleteket is szeretne megtudni az ***sshd*** többfolyamatú, privilégium-szeparált módjáról. Az ilyen módban futó ***sshd*** démon létrehoz egy privilegizált monitorfolyamatot, amely egy újabb ***sshd*** folyamatot indít a felhasználó jogosultságaival, és végül ez utóbbi indítja a parancssori értelmezőt. A privilégiumszeparáció a ***chroot***-tal valósul meg és a */var/empty* könyvtárra korlátozódik. Pontosan

ugyanarról van szó, mint a jogosultságok szétválasztásánál a ***chroot*** használatakor. A hallgatózó démon számára védelmet biztosít memóriatúlcímzés vagy más hasonló támadási kísérlet esetén. A /var/empty könyvtár legyen üres, tulajdonosa legyen a root, és a csoport, illetve a többi felhasználó számára ne legyen írható.
A privilégiumszeparáció az ***sshd*** felhasználó nélkül nem működik, és azoknál a rendszereknél, amelyek nem támogatják az ***mmap***-et vagy az ***anonymous*** memória elosztását, a tömörítést nem szabad engedélyezni.
Tegyük fel, hogy csak néhány felhasználónk van, akik ***SSH***-val szeretnének a gépre bejelentkezni, és az adminisztratív feladatok ellátását a belépés után a `su` vagy `sudo` parancsokkal végezzük el. Vegyük most azokat a szkripteket, amelyekkel az ***SSH***-t futtató kiszolgálókat próbálják megtalálni. Lehet, hogy a rendszer összes portját végigpásztázzák, áldozatot keresve, de még valószínűbb, hogy csak egyetlen port, az ***sshd*** esetében a 22-es lesz a célpont, és ha nyitva van, itt folytatódik a támadás. A kevés felhasználót kiszolgáló ***sshd*** démonok esetében tehát célszerű megváltoztatni az alapértelmezett 22-es portot, alaposan megtréfálva ezzel a szkriptek jelentős részét. Tűnjön bármily egyszerűnek ez a kis változtatás, ez fogja a nyers erővel elkövetett támadásokat a legdrámaibb mértékben visszaszorítani (*4. Lista*).
Mennyi idő kellene a világ leglassabban gépelő emberének, hogy bepötyögje a jelszót? Nyolc karakteres jelszó esetén 8 másodperc? Adjunk neki karakterenként még egy másodpercet, és legyen összesen 16 másodperc. Számoljunk hozzá újabb 20 másodpercet a hálózati késleltetés miatt, ami már több mint fél perc – igazán nagyvonalú ajánlat egy jelszó begépelésére. Az ***sshd_config*** alapértelmezett `LoginGraceTime` direktívája gyakran ezt is felülmúlja, az akár 2 percet is elérő értékével. Kérdezzük meg magunktól, mennyi ideig gondolkodnánk, hogy beengedjünk-e valakit, miután az ajtón kopogtatott? Az esethez egy másik speciális direktíva is kapcsolódik, a `MaxStartups`, amely komoly fegyver lehet a ***SSH***-portpásztázókkal szemben. Ez korlátozza az egymást követő sikertelen bejelentkezések számát. Figyeljünk arra, hogy ez engedély nélküli kapcsolatokat is jelent. Ha egy szkript nyers erővel próbál a kiszolgálóra betörni, és képes a bejelentkezési folyamatok elindítására, akár több száz folyamat is futhat párhuzamosan. Ökölszabály szerint, a `MaxStartups` értéke legyen a felhasználók számának a harmada, de nem több mint 10.

4. Lista Az SSH alapértelmezett portjának megváltoztatása

```
# Az OpenSSH-ban található sshd_config fájl
# kapcsolóinak esetében az a stratégia, hogy a
# kapcsolók az alapértelmezett értékkel szerepeljenek,
# amikor csak lehetséges, ugyanakkor megjegyzésben.
# Az alapértelmezett érték megváltoztatásához távolítsuk
# el a megjegyzésjelet és adjunk új értéket.
Port 13
Protocol 2
```

A változtatások aktiválása

Valójában nem sok mindenre kell figyelni a fentebb bemutatott változtatásokhoz. Az alapértelmezett beállítások csak útmutatásul szolgálnak, a felelősség a rendszergazdát terheli, hogy ezeket biztonság fokozása érdekében megváltoztassa. Mielőtt nekifognánk, indítsunk el egy második ***sshd*** démont egy másik porton, hogy legyen hol visszatérnünk, ha kívülről magunkra zártuk az ajtót. Az is előfordulhat, hogy a megváltoztatott beállítások miatt az sshd nem tud újra indulni. Ekkor az

```
ssh -p <másik port>
```

paranccsal egy második ***sshd*** démon is elindítható, amely kihúz bennünket a bajból. Végül is távolról dolgozunk, és ha elveszítjük a kapcsolatot, már csak a helyi konzolon köszörülhetjük ki a csorbát. Bár triviálisnak tűnik, mégis érdemes megemlíteni, hogy vagy `SIGHUP` szignált kell generálnunk, vagy újra kell indítanunk az ***SSH***-démont a változtatások életbe léptetéséhez. Továbbfűzve a másodlagos port gondolatát, olyan szkriptet is végrehajthatunk az indulásnál, amely egy második ***sshd*** démont is elindít olyan beállítással, hogy csak egy megadott felhasználó léphessen be egy megadott számítógépről. Ezzel akkor is biztosíthatjuk magunknak a bejutást egy biztonságosnak ítélt távoli gépről, ha az elsődleges ***sshd*** démon felmondja a szolgálatot.

A hajsza vége

Heroikus történetünkben a ***Linux*** rendszergazdák állnak szemben a világgal. Az egyik oldalon állunk mi, számítógépünkhöz hozzáférést biztosítva, a másikon pedig a hálózati kapcsolattal rendelkezők világa. Sokan közülük szeretnének bejelentkezni a gépünkre. Az ***SSH***-démon tulajdonságait kiaknázó eszközök gyakran az alapbeállításokat használják.
A célpontot rendszerint a fa alsó ágain lógó gyümölcsök testesítik meg, és a rendszergazda feladata, hogy ezekről tudjon és eltávolítsa, mielőtt illetéktelen kezekbe kerülnek. A gyenge védelemmel ellátott számítógép nem csak önmagára jelent veszély, hanem a hálózatba kapcsolt gépek millióira, milliárdjaira is. Az ***OpenSSH*** kiváló eszköz a távoli hozzáférés biztosításához. Leginkább egy állítható villáskulcshoz hasonlítható. Ahogy a kulcsot is állítani kell a különféle felhasználási céloknak megfelelően, az ***OpenSSH*** alapbeállításainak hangolása is szükséges a távoli hozzáférés lehetőségeinek legteljesebb és legbiztonságosabb szolgáltatásához.

Linux Journal 2006., 143. szám

Phil Moses napjait Linux rendszerek kezelésével tölti a Scripps Oceanográfiai Intézet Fizikai Oceanográfiai Kutatási Részlegén, szabadidejében pedig a világ kevéssé látogatott tájain barangol. Philnek a philmoses@cox.net címre lehet levelet küldeni.