Egyéni módszerek a hálózati biztonság növelésére

A csapdaszámítógép egy olyan hagyományos, hétköznapi, internetre csatolt gép, melyet úgy konfigurálunk, hogy minden azon keletkező eseményt a lehető legjobban meg tudjunk figyelni. A rendszerünk alapvető célja az esetleges támadó viselkedésének megismerése és az intranetünk veszélyeztetett részeinek azonosítása. A csapdaszámítógép segítségével képet kaphatunk biztonsági rendszerünk helyzetéről, és információt arról, hogy mekkora támadói kör és milyen módszerekkel próbál behatolni védett hálózatunkba.

Az internetes biztonsági megoldások egyre összetettebb formát öltenek. A biztonsági eszközök, szoftver- és hardvermegoldások, tűzfalak és egyebek viszont még a nagy távközlési cégeknek is észrevehető költséget jelenthetnek. Bár általánosan igaz, hogy egy hosszasan, tudományos alapokon kifejlesztett célberendezés hatékonyabb, ám ez önmagában a biztonságot még nem szavatolja. Hasonlítsunk össze egy drága, ám körültekintően kifejlesztett gyári megoldást egy intuitív módon, saját erőből létrehozott védelmi alrendszerrel.

A gyári megoldás előnyei:

- hosszas kutatás eredményeképpen jött létre, sokféle támadásnak ellenáll,
- a gyártó segíti az üzemvitelt, szinte azonnal megjelenik a hibaelhárító,
- a rejtett hibák hamarabb kiderülnek,
- egy jó nevű cég drága készüléke megnyugtató megoldás lehet a vezetők szemében (noha tudjuk, hogy ez nem feltétlenül jó politika),
- széles funkcionalitás, beállítási lehetőségeket építenek be a különböző igényekhez szabva.

A gyári megoldás hátrányai:

- a hosszas kutatás nem garancia, és mivel általánosan elterjedt eszközről van szó, egy napvilágra kerülő hiba sokkal jobban veszélyezteti rendszerünket, mint saját szoftverünk rejtett hibája,
- drága,
- nem mindig ismert pontosan, hogy mi ellen véd, és mi ellen nem,
- a sok különböző funkció megcélzása miatt lehet, hogy egyes részeket a fejlesztők elhanyagoltak.

A saját gyártású megoldás előnye:

- kifejlesztése jelenleg hazánkban a gyárihoz képest sokkal olcsóbb lehet,
- bevált megoldás esetén akár piaci bevezetés is lehetséges,
- pontosan definiált feladatot lát el,
- a biztonságot egy jól körülhatárolt területen ugyanúgy növelheti, mint egy gyári megoldás.

A saját gyártású megoldás hátránya:

- auditálás nélkül nem garantálja a biztonságot,
- a fejlesztők korlátos lehetőségei miatt a komplexitása alacsonyabb,
- nehéz elfogadtatni a vezetőkkel, hogy egy viszonylag olcsó megoldásra érdemes költeni és abban megbízni.

Pro és kontra rengeteg érv hangozhatna még el, véleményünk ezzel szemben a következő: a gyári megoldásokat mindig ki kell egészíteni saját gyártású megoldásokkal. Miként az életben bárhol, általában nem tökéletes az előre gyártott klisére tervezett megoldás. Az egyéni, problémához-rendszerhez igazított megoldás hatékonyabb.

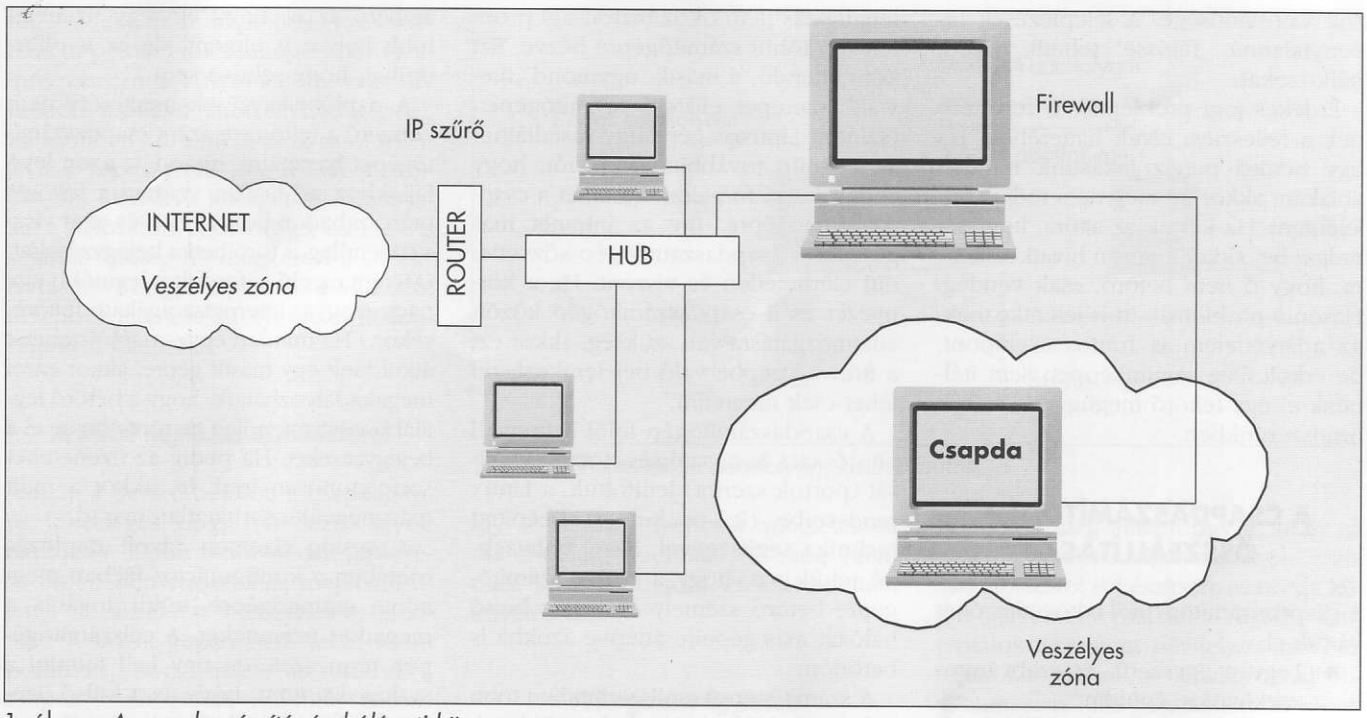
A házi vagy vállalkozóval történő fejlesztés azért is hozhat megbízhatóbb eredményt, mert egy rendszer beállításához lényegesen kevesebb tudás és körültekintés is elegendő lehet, mint egy rendszer létrehozásához. A rendszer létrehozásakor rákényszerülhet a tevékenység összes szereplője, hogy egészen pontosan tisztában legyen a létrehozott megoldás működésével és buktatóival. Sajnos a saját fejlesztéseknek és a fejlesztőknek még nincs igazán hitele ezen a téren, de véleményünk szerint ez változni fog.

Mindenképpen érdemes saját erőből megvalósítani a nagyon testre szabott, továbbá a kis részfeladatra koncentráló megoldásokat. Ha egy feladatra létezik konkrét, teljes mértékben igazodó, ellenőrzött-auditált eljárás, akkor nem célszerű saját fejlesztésbe kezdeni, mert az kevesebb támadás ellen nyújthat védelmet.

Saját védelmünk kiválasztásának első lépése a stratégiai jelentőségű pontok megkeresése és ezek megvédése. Az esetleges támadásokat itt kell azonosítani, és ezután a kezelésére rutineljárásokat kidolgozni.

A következő pontokra majdnem mindenhol lehet saját megoldást készíteni:

- naplófájlok analízise, kivonatolása, saját szempontjaink szerint,
- felhasználók tevékenységének vizsgálata, egyes pontokon a megszokottól eltérő különleges naplózással és védelemmel,
- a lehetséges külső támadók azonosítása és ideiglenes kitiltása,
- támadás, támadássorozat módszereinek felismerése,
- riasztási technikák, támadások rutinszerű kezelését segítő eljárások készítése,
- saját fejlesztésű szoftverek vizsgálatára vonatkozó eljárások és segédprogramok (saját magunk támadása auditálás céljából).



1. ábra *A csapdaszámítógép hálózati környezete*

A CSAPDASZÁMÍTÓGÉP

A hálózati biztonságtechnikában használni lehet az életből vett példákat. A védekezés, megelőzés módszereit egyértelműen átvettük: létrejöttek az azonosítási eljárások, szűrések, tűzfalak. Az immunrendszer erősítésének funkcióját látja el egy átgondolt biztonsági politika kidolgozása, a lehetséges támadások elleni védekezés ellenőrzése, az auditálás. Az eszközök újabb generációjának fejlődésében viszont még csak a kezdeti lépések figyelhetőek meg. Noha ezek is jól kidolgozott, nagy rendszerek, de az Intrusion Detection Systemek *(IDS)* még hatalmas fejlődési lehetőség előtt állnak. Az IDS rendszerek figyelik a hálózat vagy a cég informatikai rendszerét, és azonosítják a lehetséges betörést, az őrszemek szerepét vették át. [2]

A csapdaszámítógép létrehozásakor az vezérelt bennünket, hogy a lehetséges elkövetőket hatékonyan meg tudjuk figyelni. Szükségesnek látszik, hogy pontosabb képet alkothassunk a számítógép-feltörők hozzáértéséről, tevékenységéről, logikájukról. Ennek megfelelően a jelenlegi számítógépes infrastruktúránkba kívántunk behelyezni egy olyan improduktív számítógépet, amelynek alapelveit a következőkben határozzuk meg.

1. A csapdagép az internet felől szabadon elérhető, ne szűrjük az adatokat a gép felé, de oldjuk meg saját hálózatunk védelmét. Egy kísérletezgető, aki az internetes rendszerbe próbál betörni, ezzel a rendszerrel is próbálkozik. Próbálkozásai sikertelenek, de mi tudni fogjuk, hogy valaki próbálkozott, és azt is, hogy mennyire makacsul. Ezt egy szűrő funkcióval ellátott tűzfal nem biztos, hogy megteszi.

2. A csapdaszámítógépen fontosnak vélt, ám valójában nem fontos információk vannak. Ha egy betörőnek sikerül feltörnie ezt a számítógépünket (amely talán – célszerűen – a legvédtelenebb a hálózatunkban), akkor tudni fogjuk, hogy a többi számítógépünket is rövidesen megtámadhatják, és képünk van a támadás módszereiről, azonnal riadót rendelhetünk.

3. Ha van egy csapdaszámítógépünk, amely mindent naplóz és mindenre riasztást küld, akkor célzott kísérleteket is végezhetünk. Rögtön észrevehetjük, hogy lehallgatják hálózatunkat, ha titkosítatlan névvel és jelszóval néha személyesen vagy automatikusan belépünk a csapdára. Ezután az esetleges lehallgató személy kódunk, azonosítónk birtokába juthat, de amint ezt kipróbálja erről értesülünk a riasztásból. Konkrétabb példán: XY internetszolgáltató sok számítógép tárolását vállalja. Nem tudja azonban, hogy egy-egy számítógép nem hallgatja-e le egy adott HUB-ra csatolt többi számítógépet, vagyis egy bérlőjének gépe nem veszélyezteti-e saját hálózatát vagy a többi gépet. Ha ugyanerre a HUB-ra ráköti a csapdaszámítógépet, majd arra célzatosan kódolatlan jelszóval vagy más módon esetlenül belép, az egy esetleges támadót arra ingerelhet, hogy maga is próbáljon belépni a csapdára. A csapda természetesen azonnal riaszt, mi pedig biztosak lehetünk, hogy valamely ügyfelünk számítógépével gond van, erősen veszélyeztetheti az egész rendszert.

4. Egy betörő minden cselekedetét meg tudjuk figyelni a csapdában. A számítógépek feltörői gyakran átlépnek egyik gépről a másikra, gyakran használnak ki egy feltört számítógépet újabb gép feltörésére, és összevissza mozognak a feltört számítógépek között. Ha egy betörő csapdaszámítógépünkről további gépeket tör föl, azok tulajdonosait azonnal figyelmeztethetjük a betörésre. Ha már régebben feltört számítógépekre megy vissza, esetleg ott elhelyezett kiskapukat próbál kihasználni, akkor birtokában leszünk az általa feltört gépek listájának (vagy egy részének), és a használt kiskapunak is. Ez utóbbi például rendkívül hasznos lett volna a legutóbb feltört nagy internetszolgáltató számára, amelynél szintén elképzelhető, hogy kiskapu volt a rendszerében.

5. Betörőnk cselekedetei közben értékes információkat leplezhet le magáról. Aláírt (névvel, becenévvel ellátott) levelet írhat az egyik feltört gépről, vagy beléphet valamely azonosítást követelő szolgáltatásba. Ezek alapján elképzelhető, hogy névvel vagy akár lakcímmel tudjuk megcímkézni a betörőt, és ha tényleg feltört valamilyen fontos rendszert (vagy akár a csapda feltöréséért is), felelősségre vonható. Néhány, ilyen módon azonosított és pellengérre állított betörő példájából okulva talán mérsékelhető a támadá-

sok valószínűsége. A leleplezések bizonytalanná, „félőssé" tehetik a próbálkozókat.

Érdekes jogi problémák is felvetődnek a fejlesztési elvek hátterében. Ha egy betörő benéz lakásunk minden ablakán, akkor mi még nem tudjuk feljelenteni. Ha kiírjuk az ajtóra, hogy fáradjon be, akkor jogosan hivatkozik arra, hogy ő nem betörő, csak vendég. Hasonló problémák itt is jelentkeznek, az adatvédelem is fontos szempont, de erkölcsileg semmiképpen sem ítélnénk el egy feltörő megfigyelését saját rendszerünkben.

A CSAPDASZÁMÍTÓGÉP ÖSSZEÁLLÍTÁSA

A csapdaszámítógéptől a következőket vártuk el:

- Legyen egyszerű, és szabványos eszközökre épüljön.
- Ne veszélyeztesse a meglévő számítógépes hálózat biztonságát, azaz magán a csapdaszámítógépen kívül más számítógépekre ne jelentsen veszélyt.
- A betörő ne, vagy csak nehezen vegye észre, hogy betörése közben figyelik.
- Minden kiértékelhető információt rögzítsünk a betörésről. Legyen tehát strukturált az információ, de ha valami részleteiben érdekesnek bizonyul, akkor az is analizálható legyen.
- A rendszer legyen könnyen kezelhető és átalakítható.

A felépített rendszerünk a következőket tartalmazza.

A rendszert a Debian GNU Linux disztribúció „slink", 2.1-es verziójából hoztuk létre, mely 1999 márciusában jelent meg (azaz a rendszerbeállításkor 7 hónapos volt). Az operációs rendszert egy viszonylag egyszerű és olcsón beszerezhető számítógépre telepítettük (Intel Celeron 366 MHz, 64 Mbyte RAM, 6.4 GB HDD, ...)

Bárki, aki a csapdagépet feltöri, a gép hálózati csatolóján átmenő adatokhoz hozzáférhet (sniffelheti). Így igen fontos, hogy a gép vagy egy kapcsolt hálózati szakaszon legyen, vagy – mint megoldásunkban – a gépet egy másik gépen át tettük elérhetővé *(1. ábra)*. Rendszerünk valójában két gépből állt, a csapdaszámítógépből és a hozzá kapcsolódó tűzfal funkciót ellátó gépből. Ezzel a megoldással elértük, hogy a csapdaszámítógép hálózati csatolóján az intranet más forgalma nem megy át. Így – ha nem használják az csapdagépet más célra – az ottani lehallgatás nem okoz biztonsági problémát a többi számítógépre nézve. Ezt könnyítendő, a másik, úgymond „firewall" szerepet eljátszó számítógépen (szintén Linuxos gép) úgy installáltuk az IP szűrt továbbítás funkciót, hogy csak a külső forgalom kerülhet a csapdaszámítógépre. Így az intranet más gépeiről a csapdaszámítógép közvetlenül elérhetetlen és viszont. Ha a környezet és a csapdaszámítógép között adatmozgatásra van szükség, akkor ezt a firewall gépbe való bejelentkezéssel lehet csak megtenni.

A csapdaszámítógép felől a firewall gép összes nem szükséges szolgáltatását (portok szerint) letiltottuk, a Linux rendszerbe (2.2-es kernel) beépített technika segítségével. Ezzel nehezebbé tettük azt, hogy a csapdaszámítógépre betörő személy esetleg a belső hálózat más gépeire átlépve azokba is betörjön.

A számítógépes rendszerünkben több helyen szűrőfunkciót helyeztünk el. Ezeknek nem célja a csapda felé irányuló forgalom korlátozása, hanem a hálózatban lévő többi számítógép biztonságát is érintő támadásokat kell kiküszöbölniük. A szűrők tehát a csapda felé minden szolgáltatást továbbítanak, és csak a belső hálózatot védik.

A naplózás beállítása a következő lépés, miután a csapdaszámítógép már fogadni és továbbítani tudja az adatokat. Feladata az információtárolás pontos feltételeinek megteremtése. A Linux alapkiépítésében is tárol információkat arról, hogy mi történt a rendszerrel. Az adatokat a központi *syslogd* nevezetű daemon (kiszolgáló-felügyelő program) menti.

A syslogd pontos információkat kap minden belépésről, az egyes szolgáltatások üzeneteiről. Ezeket az üzeneteket úgynevezett Syslog facilityk szerint csoportosítva kapja, amelyeken belül további csoportosítás is lehetséges. Azt, hogy melyik naplófájlba pontosan mit kell elmenteni, a syslogd egy konfigurációs fájlból olvassa ki. Elképzelhető az is, hogy egy-egy üzenetet több helyre is elment, de az is előfordulhat, hogy sehova sem.

A naplóbejegyzések mentésére nem célszerű a feltörésre szánt csapdaszámítógépet használni, hiszen az azon levő fájlokhoz a hívatlan vendég a feltörés után szabadon hozzáférhet és akár visszamenőleg is törölheti a bejegyzéseket. (Miként ez elő is fordult a legutóbb egy nagy magyar internetszolgáltató feltörésekor.) Ha minden egyes naplóüzenetet átküldünk egy másik gépre, akkor ezzel megakadályozhatjuk, hogy a betörő legalább visszamenőleg ne tüntethesse el a bejegyzéseket. Ha pedig az üzeneteket sorfolytonosan írjuk le, akkor a múlt már megváltoztathatatlan marad.

A syslog daemon távoli naplózás esetében a konfigurációs fájlban megadott számítógépre küldi tovább a megadott üzeneteket. A célszámítógépen természetesen úgy kell futtatni a syslog daemont, hogy az a külső üzeneteket fogadja *(2. ábra)*.

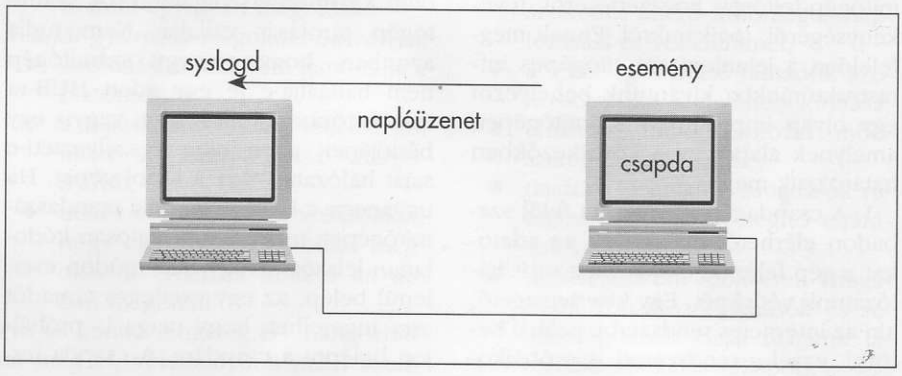
A problémát az okozza, hogy ha megnyitjuk ezt a lehetőséget, akkor elvileg bárki a világból teleszemetelheti gépünket hamis üzenetekkel.

Másik megoldandó kérdés: a távoli naplózás tényét a betörő hamar észreveheti. Erre két lehetősége is kínálkozik, az egyik a hálózati forgalom lehallgatása, a másik a syslog konfigurációs fájljának egyszerű megtekintése.

A syslog daemon forrásprogramjának módosításával az újrafordítás után már nem a szokásos helyről veszi a konfigurációs információkat, hanem egy másik fájlból. Ekkor a betörő nyugodtan hiheti, hogy az eredeti helyen levő konfigurációs fájl tartalmazza az igazi bejegyzéseket. Ügyelni kell arra, hogy ha ezt a fájlt mégis editálni kezdik, akkor nem látják a naplózás megváltozását. Ezért a legcélszerűbb az eredeti fájl felhasználása, és ehhez egy további kiegészítő fájl használata.

A másik probléma, a lehallgatás nehezen védhető. Egyik megoldásunk lehetne, hogy nem a hálózaton keresz-

2. ábra *Távoli naplózás*



tül küldjük át a naplóbejegyzéseket, hanem például a számítógép párhuzamos portján át (nyomtatóra vagy ily módon csatlakoztatott másik gépre). Ez azonban körülményes és nem sokat javít a rendszeren. Egy másik megoldás, ha titkosítva küldjük át az adatokat a hálózaton. A betörő ez esetben is sejtheti – főleg ha megkeresi a „jelek" forrását –, hogy naplózásról van szó, de ennek ellenére a folyamatot biztonságosabbá teheti.

A távoli naplózás új módszerei

Miután a betörő bejutott a csapdagépre, lehetséges, hogy felfedezi a távoli naplózást. Ekkor betörésének leplezése érdekében elronthatja, meghamisíthatja az átküldött bejegyzések tartalmát. A távoli gépen nem lehet tudni pontosan, hogy mikor kezdte el a csapdaszámítógépen működő betörő a naplóbejegyzéseket meghamisítani. A syslog-ng *(syslog Next Generation)* fejlesztésekor ezt a problémát küszöbölik ki.

A megoldáson persze sokan mások is dolgoztak.

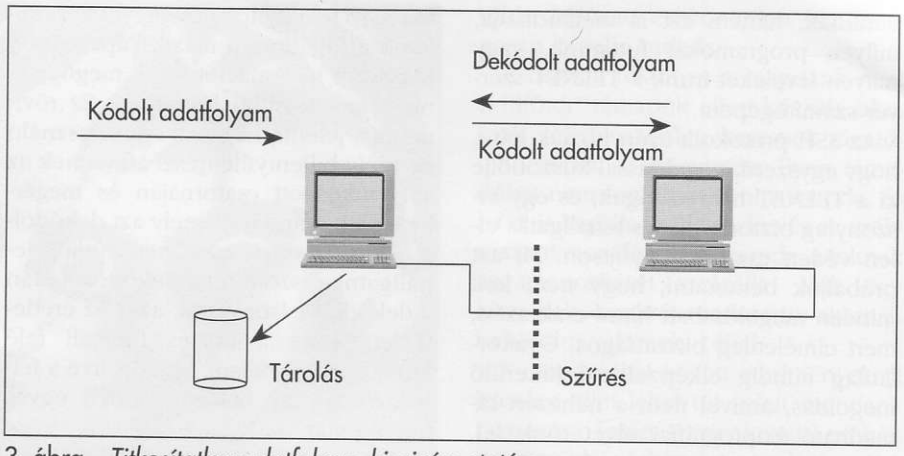
Egy egyszerű eljárást mutat be a biztonságosabb naplófájl-kezelésre Kargieman és Futoransky a PEO-1 protokoll segítségével [3]. A módszer egyszerű hash-függvényen alapul. (A hash-függvény olyan egyirányú függvény, ami egy nagyobb méretű adathalmazról egy kisebb méretű adathalmazra képez le; hash-függvénnyel leképzett „lenyomatot" használnak például digitális aláírás bemeneteként.)

A megoldás lényege a következő: a PEO-1 a naplóbejegyzéseket egy ellenőrző kóddal látja el, az új ellenőrző kód értéke az előző kód és a mostani naplóbejegyzés adata páros alapján gyártott lenyomat.

Ezzel ugyan még nem oldják meg teljesen a problémát, mert a betörő a csapdaszámítógép memóriájához is korlátlanul hozzáférhet, és így gyakorlatilag minden adat a rendelkezésére állhat.

A jelenlegi operációs rendszerekben, melyben léteznek kiemelt prioritással rendelkező eljárások, melyek korlátlanul hozzáférhetnek a rendszer erőforrásaihoz – így a teljes memóriához is – nem létezhet olyan kriptográfiai algoritmus, amely teljes bizonyossággal meg tudja őrizni az információk titkosságát, és amely algoritmus nem függ a számítógép szempontjából külső beviteli egységről származó információktól. (Ezen tételünket itt nem bizonyítjuk.)

A PEO-1 ha nem is lehetetlenné, ám igen körülményessé teszi a támadást.



3. ábra *Titkosítatlan adatfolyam kiszivárogtatása*

Így a rendszerüzenetek megfelelően menthetőek, azonban ez még igencsak kevés a betörő cselekedeteinek követésére.

A forgalom figyelése

A csapdaszámítógépet a firewall-gépen keresztül tettük elérhetővé, így minden forgalmat, ami a csapdaszámítógép felé megy vagy onnan jön, a firewallon megfigyelhetünk. Megfigyelésre természetesen használhatunk nagy IDS rendszereket is. A legtöbb ilyen rendszer jelezné a betörés tényét, ugyanakkor a betörő további tevékenységének megfigyelésére már nem annyira alkalmasak. Ezen kívül nincs lehetőségünk az IDS rendszer által megfigyelt számítógép felé irányuló, illetve az onnan induló rejtjelezett adatfolyamok megfigyelésére sem. Ilyen például az SSH *(secure shell)* közismert protokollja.

Sniffit

A csapdaszámítógép hálózati forgalmából bizonyos szolgáltatások adatait a *sniffit* nevű programmal egy megfelelő alkönyvtárba rögzítjük. Ez a program a firewall Ethernet kártyáját az úgynevezett „promiscuous" módba kapcsolja. Így minden Ethernet-csomag feldolgozásra kerül, nemcsak azok, amelyek célpontja az adott számítógép. A sniffit a TCP kapcsolatokon átmenő adatokat fájlokba menti, melyek nevei tartalmazzák az adott kapcsolatban résztvevő két IP címet és a portcímeket.

Ezzel a megoldással (ha a sniffit bírja a hálózati forgalmat és nem hagy ki csomagokat), naplózni tudjuk a csapdaszámítógépre tartó TELNET adatfolyamokat, az FTP kapcsolatokon átment parancsokat, az átmenő leveleket, WWW kéréseket és pár egyéb dolgot. A sniffitet be tudtuk még állítani úgy is, hogy az egyes speciális szolgáltatásokról összesítő táblázatot is készítsen, azaz egy külön fájlba menti például a TELNET kapcsolatokon átment név-jelszó azonosítópárokat. Ezzel a lehallgatással továbbra sem lesz használható naplónk sem az SSH protokollon, sem bármely SSL-t használó szolgáltatáson átment adatfolyamokról.

A betörő megfigyelésének legfontosabb lépése a shell parancsok követése. Ez a TELNET kapcsolat lehallgatásával már megvalósult, azonban ma már a TELNET protokoll helyett majdnem mindenki a titkosított SSH szolgáltatást használja, amely egy lehallgató számára, így itt számunkra is önmagában csak egy zagyva jelfolyam.

Az SSH naplózása

Az internetes protokollok, hálózati alkalmazások általában két fő programrészből állnak, melyek között adatátvitel zajlik. Ez a két programrész valójában egy-egy önálló program, az egyik az ügyfél számítógépén fut (ez a kliens), a másik a kiszolgáló számítógépen (ez a szerver daemon). Az SSH protokollt használó hálózati alkalmazás kliens részét *ssh-nak*, míg szerver oldali programját *sshd-nek* hívják.

Az SSH és a TELNET alkalmazásokat arra hozták létre, hogy egy számítógép segítségével egy másik számítógépen parancsokat adhassunk ki, távoli szöveges programokat futtassunk, és ezek kimenetét, eredményét saját számítógépünkön figyelhessük meg.

A TELNET és az SSH voltaképpen arra épül, hogy minden billentyűleütést, amelyet a felhasználó saját számítógépén tesz, átküld a hálózaton keresztül. Az átment adatok lehallgatása tehát azért veszélyes, mert a lehallgató nemcsak birtokába kerülhet belépési kódunknak, azaz nevünknek és jelsza-

vunknak, hanem azt is megtudhatja, milyen programokat futtatunk, vagy milyen leveleket írunk a TELNET szerver számítógépen.

Az SSH protokollt azért hozták létre, hogy egyszerű megoldással küszöbölje ki a TELNET hiányosságait, és egy viszonylag biztonságos és lehallgatás ellen védett csatornát nyújtson. Mi azt próbáljuk bemutatni, hogy nem kell minden megoldásban hinni csak azért, mert elméletileg biztonságos. Gyakorlatilag mindig elképzelhető kikerülő megoldás, amivel nem a nehezen támadható kriptográfiai elvet törik fel, hanem egy kiskapu az egész algoritmust kerüli meg.

Az SSH többfajta titkosítási és authentikációs eljárást tartalmaz. Authentikálásnál általában kétféle módszert választhatunk: azonosítást nyilvános kulcsú rejtjelezés segítségével, illetve a hagyományos, jelszó alapú azonosítás módszerét.

Ahhoz, hogy a titkos jelfolyamot dekódolni tudjuk, lehetőségünk van átírni az SSH protokollt megvalósító programokat, hogy a firewallon kitalálható legyen a kapcsolat alatt használt összes kulcs. Ehhez az adott számítógép (a mi feladatunkban a csapdagép) nyilvános és titkos RSA kulcsát is át kell vinni a firewallra. Ezután az egész kapcsolat alatt fennálló titkos adatfolyamot vissza kell fejteni a korábban megfigyelt, kulcscsere során létrejövő kapcsolat-kulccsal. Ez az út bonyolult, ezért nem teljesíti a csapda specifikációjában elvárt egyszerűséget.

Az SSH és a TELNET is egy shellt (parancsértelmezőt) fog futtatni, melynek kódját módosíthatjuk úgy, hogy minden rajta átmenő adatot lássunk. Találhatunk is már megírt programcsomagot ilyen célokra, például *ttysnoop*. A dolog csak azért problémás, mert ezeket az adatokat szeretnénk biztonságosan elmenteni, például a firewall gépre. Ekkor pedig a lehallgatott adatokat a syslogdnek kellene küldenünk, ám több új probléma merül fel: a syslogd naplóbejegyzések feldolgozásánál és a mentésnél is pontosan kell követni, melyik üzenet melyik bejelentkezéshez, melyik éppen futó shellhez tartozik. A vezérlőkarakterek átvitele is problémákat jelent.

Ennél sokkal egyszerűbben megvalósítható az ssh és az sshd programok, azaz az SSH kliens és SSH szerver átírása. Mindkét programot forráskódjában úgy módosítottuk, hogy azok egy-egy kapcsolatnál ne csak a megszokott titkos kapcsolatot, hanem kapcsolatonként két, nem titkos kapcsolatot is nyissanak meg. A nem titkos kapcsolatot az általunk átírt ssh a firewall számítógép felé nyitotta. Az egyik kapcsolaton a fogadott, a másik kapcsolaton a küldött információt lehet megfigyelni, nem titkosított formában. Ez röviden azt jelenti, hogy az ssh-t használó személy billentyűleütései átmennek az SSH titkosított csatornáján és megérkeznek a csapdára, amely azt dekódolja. Ezt titkossága miatt nem tudjuk lehallgatni, viszont mi a dekódolás után a dekódolt jelsorozatot, azaz az eredeti leütéseket a köztes firewall felé küldjük el rögzítésre, így rögzítve a felhasználó vagy betörő minden egyes mozdulatát.

Megjegyezzük: ha nem akarjuk, hogy a betörő lehallgatás esetén rájöjjön, hogy mi is lehallgatjuk őt, akkor egy egyszerűbb vagy bonyolultabb titkosítással ezt a nem titkos kapcsolatot is védhetjük.

A nem titkos kapcsolaton átment adatok elmentéséről a firewallon rövid perl programok gondoskodnak. Az sshd az authentikáció során megkapott jelszót és felhasználói azonosítót is továbbítja a nem titkos kapcsolat irányába, mivel az egyébként nem lehetne megfigyelhető. Az ssh programot természetesen csak binárisan raktuk fel a csapdaszámítógépre, és mint sejthetjük, nem sok betörő fogja ellenőrizni, hogy a rendszerben levő bináris fájlok az adott szituációban hitelesek-e avagy sem. Így legalább az első akcióig minden bizonnyal titkos tud maradni tevékenységünk.

Az ssh programok átírásával mód van a betörő további megfigyelésére is.

Feltételezhetjük, hogy a betörő (azért, hogy magát elrejtse, vagy egyéb okból, esetleg véletlenül is) a feltört csapdaszámítógépről fog újabb számítógépekre továbbmenni. Mi ezeket is pontosan meg tudjuk figyelni.

A dolog voltaképpen egy rejtett biztonsági probléma: az SSH protokoll mindig csak a legközelebbi szerverig titkosít, ha onnan újra továbbmegyünk, akkor a már dekódolt adatfolyam kódolódik újra *(4. ábra)*. Az SSH protokoll tehát többféleképpen köthető „sorba". Ha *A* gépről *B* érintésével akarjuk elérni *C* számítógépet, akkor ezt úgy célszerű megtenni, hogy B minden egyes hozzá érkező adatot úgy továbbít *C-hez,* ahogy megkapja, mérnöki szóval rövidre zárja, mintegy galvanikus kapcsolatot biztosít. Ha azonban *B* számítógép ezt a jelfolyamot dekódolja és azt újra kódolva küldi tovább, akkor nem viselkedik tökéletesen ugyanúgy, ahogy az előbbi esetben, azaz egyfajta transzformátor funkciót lát el. Ez a különbség a biztonság szempontjából igen fontos, mert ilyenkor *B* számítógépen közvetlenül megjelenik a titkosítatlan adatfolyam.

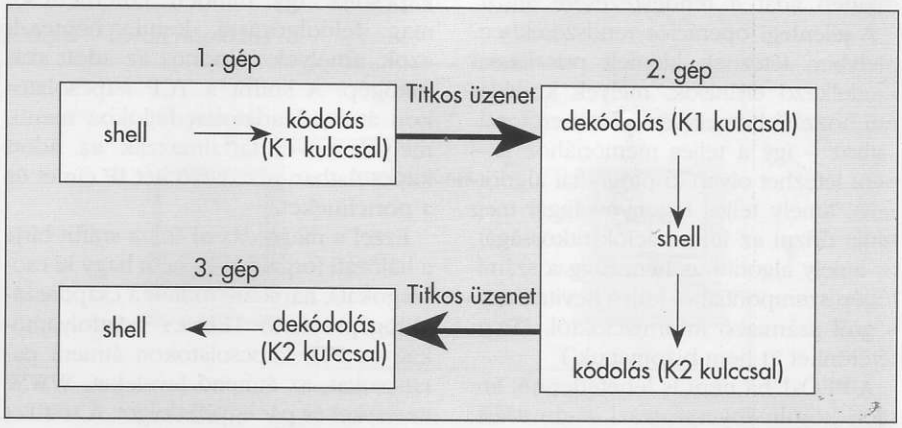
Ha ilyen esetben a középen elhelyezkedő szerver csak buta ismétlő lenne, és a távoli szerverhez menő adatokat csak egyszerűen megismételné, a kliens pedig közvetlen a távoli szervertől szerezne kulcsot a kódoláshoz, akkor a rendszer biztonságos tudna maradni. (Természetesen módot adna man-in-the-middle típusú támadásra, de ez ellen az SSH eleve védekezik a szerver RSA alapú azonosításával.)

Nagyon érdemes átgondolni tehát, hogy noha az SSH-ban jobban megbízunk mint egy titkosítatlan adatfolyamban – ennek ellenére ugyanúgy le lehet hallgatni, persze csak az egyik oldal aktív közreműködésével. A hamis biztonságérzet viszont szabadságot adhat a meggondolatlan ember kezébe, ami veszélyt jelenthet számára. Úgy gondolja, hogy biztonságban van, közben pedig nem, így viszont jóval nagyobb baj érheti, mint ha megfelelően óvatos lenne.

További teendők, a betörő csalogatása

A naplózás fő feladatait ezzel megoldottuk. Már csak egy pár apró lépést kell tenni annak érdekében, hogy pon-

4.ábra *Az SSH működési módja több gépen keresztül*



Egyéni módszerek ... *(Cakó Ferenc homokgrafikája)*

tosan tudjuk, mikor mi történik a rendszerünkön.

El kell helyezni apró trükköket:

- a belépés után automatikusan lefutó programok (shell-beállítások, például bashrc) belsejébe elhelyezhetünk figyelmeztető rutinokat, melyek e-mailben figyelmeztetik a rendszergazdát, ha valaki az adott felhasználó nevén belépett;
- megváltoztathatjuk az *rm* (fájl törlés) parancsot, hogy a fájlokat ne törölje véglegesen, csak egy ideiglenes helyre rakja át.

Ezen a pontot igen sok intuitív elem kerülhet a rendszerbe, amely minél meglepőbb helyen van elhelyezve, annál könnyebben, gyorsabban buktatja le a feltörőt.

Fontos, hogy az installált majd feltört rendszerünkben már mi sem bízhatunk, így egyrészt érdemes archiválni a rendszerünket az esetleges betörés előtt, illetve a későbbi rekonstrukció vagy változás-detektálás érdekében célszerű minden fájlról ujjlenyomatot *(fingerprint)* venni. Az [1]-ben kifejtett módon például a tripwire programmal MD5 lenyomatokat tárolhatunk a felrakott programokról, így a rendszer állapotát folyamatosan ellenőrizhetjük.

Fontos és elengedhetetlen lépés, hogy kirakjunk a belépéskor egy figyelmeztető üzenetet. Ez tartalmazza azt, hogy a rendszert csak engedélyezett felhasználók használhatják, és hogy minden billentyűleütést monitorozhatunk. Erre a jog visszássága miatt mindenképpen jobb odafigyelni, hiszen esetleg mi ugyan nem tudjuk beperelni a betörőt a feltárt tevékenységéért, de a betörő beperelhet minket az illetéktelen lehallgatásért.

Az apróbb trükkök közé tartozik még a figyelem felkeltése és koncentrálása. Érdemes a gép nevét úgy megválasztani, hogy az könnyen potenciális betörők célpontja legyen. Például egy www.torjfel.hu címet bármely betörő érdekesnek találhat.

Miután a csapda látogatója belépett, és szeretné a gépet feltörni, erre is módot lehet adni. Mivel a Debian programrendszert megfelelően frissítik, így nem könnyű olyan állapotban hagyni a gépet, hogy az jól feltörhető legyen. Erről úgy gondoskodtunk, hogy az 1999. márciusi (elég réginek számító) változatot raktuk fel, és az azóta fellelt hibák javítását tartalmazó csomagokat nem frissítettük.

Még számtalan hasonló trükköt lehet elhelyezni a rendszerben, ezek segítségével az esetleges betörő hamar „nem megengedett" jogosultságokat tud szerezni.

A csapdaszámítógép így tehát gyakorlatilag készen van. Most már csak várni kell a betörőket, és vizsgálni, mit csinálnak...

ÖSSZEGZÉS

A csapdaszámítógép egy kombinált rendszer, amely magán hordozza az IDS rendszerek sajátosságait (például a betörő megfigyelése), ám természetesen alkalmaz további jól bevált módszereket is. Ilyenek a riasztások, állományok auditálása, naplózás, a kapcsolódó firewall, szűrés stb.

A csapda használatával kapcsolatosan fontos dolog megjegyezni, hogy a csapdagép felállítása miatt az eddigi rendszerünk biztonsága is megváltozott. Úgy sejtjük, hogy nem csökkent a biztonság, (mivel megfelelően alkalmaztuk a csomagszűrés módszerét, nem alkalmaztunk máshol is használt jelszavakat stb.), arra azonban oda kell figyelni a továbbiakban, hogy a többi hálózati eszközünket biztonságosan kezelni tudjuk.

A célokat tekintve: a csapdánk nem „külföldi" támadók ellen készült, akik csak véletlenszerűen keresnek támadható gépet az interneten. Ilyen támadókat nem fog elriasztani a csapda, és ha el is kapják az illetőt, kevés az esély a szankcionálásra. A csapda azoknak az azonosítását szolgálja, akik egy kiszemelt, konkrét rendszerbe akarnak behatolni valamely céllal.

A csapda által hozott eredményekről jelenleg nem lehet túl sokat mondani: nem sokan (szám szerint ketten) próbálták meg idáig feltörni a csapdánkat. Ennek több oka van, a legfőbb ok az, hogy nem volt rajta semmilyen vonzó információ. Vonzóvá tenné a csapdát az is, ha nem magán a csapdán, csak a környezetében tárolnánk fontos információkat. Egy betörő módszeresen körüljárja célpontját, hátha egy másik, mellékes gép feltörésén keresztül jut közelebb áhított céljához. Ezért jól lehetett volna használni a csapdát egy nagyobb cég, például egy internetszolgáltató számítógépes hálózatában is.

Felmerül persze a kérdés, hogy kezdetben ugyan jó a csapda arra, hogy betörőt fogjunk, de mi fog történni, amikor már a feltörők tisztában lesznek azzal, hogy csapdával találkozhatnak? Kicsi az esély arra, hogy a csapda általánosan bevett eszközzé alakuljon, de ha a legnagyobb szolgáltatók kiszámíthatatlan módon néhány csapdát helyeznek el, akkor a feltörők mentalitása megváltozhat. Nem az elfogás lesz a csapda elsődleges célja (bár a feltörők vizsgálata mindig tanulságos lehet), az elriasztás lesz a fő cél. A betörő biztonságérzetének csökkentésével jelentős energiákat lehet lekötni, így növelve rendszerünk biztonságát.

Bencsáth Boldizsár – Tihanyi Sándor

Irodalom

[1] The design and implementation of Tripwire: A file system integrity checker. Gene Kim, Eugene H. Spafford. Technical Report CSD-TR-93-071. Purdue University, 1993

[2] FAQ: Network Intrusion Detection Systems, Robert Graham, Version 0.4.1, April 15, 1999. http://packetstorm.securify.com/docs/infosec/network-intrusion-detection.html

[3] VCR y PEO: Dos protocolos criptográficos simples, Emiliano Kargieman, Ariel Futoransky, 1995 illetve ennek módosítása (PEO revised Oct, 1998) http://www.core-sdi.com/english/publications.html, http://www.core-sdi.com/soft/vcr-peo.doc.gz,

[4] Security of the Internet, The Froehlich/Kent Encyclopedia of Telecommunications vol. 15. Marcel Dekker, New York, 1997, 231-255. http://www.cert.org/encyc_article/tocencyc.html