Operációs rendszerek egyenrangú közlésen alapuló védelme

Az internet növekedésével és elterjedésével a számítógépes biztonsági kérdések egyre inkább előtérbe kerültek. Manapság az egyik legnagyobb veszélyt számítógépeinkre a hálózati támadások jelentik. A cikkben bemutatott program éppen a hálózatot próbálja meg arra használni, hogy a kiszolgálók a támadásoknak minél hatékonyabban ellenállhassanak. A működés során a programot futtató számítógépek egy P2P hálózatba szerveződnek, amelyen keresztül megosztják az érzékelt betörési kísérletekről gyűjtött tapasztalatokat, így növelve az összes résztvevő biztonságát.

A szoftvert *Komondor*nak neveztük el, hiszen feladatkörében sok minden hasonló a házőrzésben híresen kiváló kutyafajtára. Léteztek már korábban is a hálózaton egymással kapcsolatot tartó biztonsági programok. Az itt bemutatott szoftver újdonsága az, hogy az egyes gazdagépeken futó példányok az Interneten egy *egyenrangú* (*peer-to-peer, P2P*) hálózatot hoznak létre. A szerveződés önműködő, felhasználói beavatkozást nem igényel. Ez a hálózati modell nagy stabilitást biztosít, amelyre az egyes egyedek között a tapasztalatok gyors, megbízható átadása miatt van szükség. A rendszer felépítéséből adódóan a hálózati hibák és főként a támadások miatt labilis hálózaton is működőképes marad.

Fontos megjegyezni, hogy a *Komondor* elfedni hivatott az egyes egyedek operációs rendszereinek, szolgáltatásainak biztonsági réseit, nem pedig kijavítani. Ehhez nincsen szüksége az adott biztonsági rés ismeretére. Előzetesen képes lehet bizonyos védelmet kialakítani, de csak akkor, ha valahol egy helyen történt már betolakodási kísérlet. Nem az adott rést foltozza be, hanem a konkrét támadót akadályozza meg a további tényekedésben. Ha az adott rés ismert, érdemes inkább annak közvetlen kijavításával foglalkozni.

A program első változata *Linux* rendszerre épül, *C* programozási nyelven íródott. A bemutatása után becslést adunk egy ilyen átfedő általános használhatóságára, bemutatjuk az előnyeit és a korlátait is.

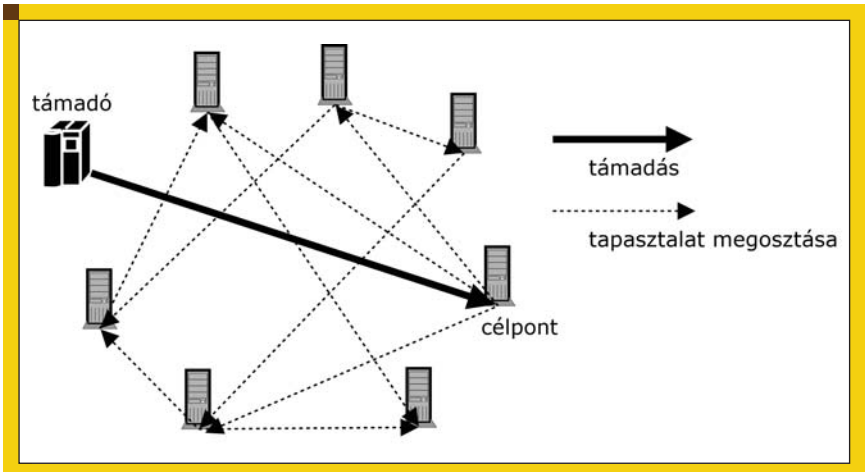
A biztonsági rések és a támadások természete

Ha a támadó egy adott számítógép *működését zavarni* szeretné, vagy *adatokat* szeretne megszerezni róla, akkor a célba vett kiszolgáló hálózati címe általa előre ismert. Ebben az esetben nyitott hálózati *kapukat* (port) keres szolgáltatások után kutatva, annak reményében, hogy biztonsági rést talál. Gyakori a *kapupásztázás* (port scan), amely a számítógépen futó alkalmazások hálózatról való megcímzéséhez szükséges kapuk teljes tartományának végigpróbálását jelenti. Ennek célja az, hogy találjon valamelyik kapun egy hibás alkalmazást, amelyet azután felhasználhat a rendszerbe való bejutáshoz. Kifejezetten erre a célra tervezett programok is léteznek. Ezek nem feltétlenül rossz szándékból íródtak, hanem saját rendszerünk biztonságának tesztelésére is alkalmasak. Ilyen alkalmazás a jól ismert *Nmap*. Biztonsági résnek számítanak a hanyag felhasználók *gyenge jelszavai* is. Ezt használja ki az ún. szótár módszer, amely arra a tényre épül, hogy a gyenge jelszavak köznevek vagy gyakori tulajdonnevek. Ezek viszonylag kis számú próbálkozással kitalálhatóak. Hibás szolgáltatáson keresztül *erőforrások* után is kutathat a támadó. Jellemzőbb ekkor az, hogy a pásztázás egy adott *IP* szám tartomány ellen irányul. A kapuszám ilyenkor kötött: például a 25-ös port, *SMTP* kiszolgáló keresése levélszemét küldés céljából. A fenti támadások bár különböző módszerekre épülnek, mégis közös vonásuk, hogy a támadó több kísérletet tesz céljának elérésére. A *Komondor* ezt használja ki: ha ugyanis valamelyik *Komondor* egyed betolakodást észlel, és ezt a tényt megosztja a többiekkel, akkor azok már felkészülve képesek várni *ugyanazt a támadót*, aki módszereinél fogva (pásztázás) nagy valószínűséggel előbb-utóbb meg is érkezik.

A kiszolgálók biztonsága

Egy számítógép soha nem lehet teljesen biztonságos. A szakirodalom meghatározza a *megfelelően képzett támadó* fogalmát, egy olyan elméleti személyt, aki szaktudásánál fogva képes felderíteni bármilyen biztonsági rést. Tudjuk, hogy nem hozhatunk létre tökéletesen hibamentes rendszert. Ennek ellenére számítanunk

kell rá, hogy bármely hibát lehetséges megtalálni, akár szisztematikusan, akár véletlenül.
Az átlagfelhasználónak sokat nem kell tennie a rendszere biztonságossá tételéhez. Értékes adatok tárolása esetén azonban mindenképpen foglalkoznunk kell ezekkel a kérdésekkel. Fontos megjegyezni, hogy minél biztonságosabb egy rendszer, annál kisebb a használhatósága a szigorú feltételek miatt. A biztonság és a használhatóság között kompromisszumot kell létrehoznunk. Egyszerű példa erre a hálózati forgalom korlátozása.



1. ábra Támadás a Komondor rendszer egyik tagja ellen

A betolakodás-észlelés fogalma

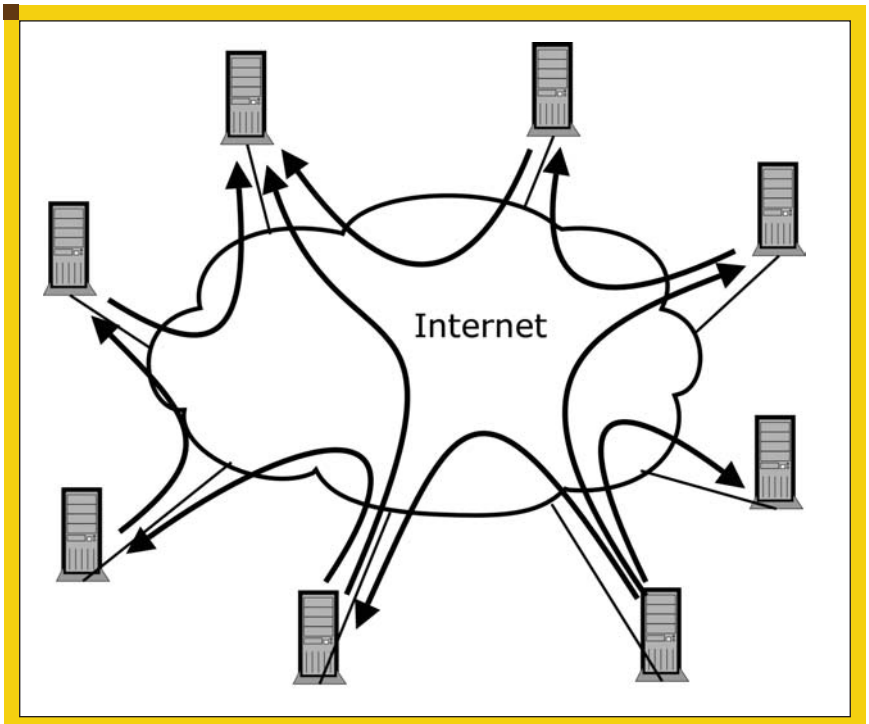
A betolakodás-észlelő technikáknak két fő kategóriája van: a ***rendellenesség-észlelés*** (***anomaly detection***) és ***visszaélés-észlelés*** (***misuse detection***).
A ***rendellenesség-észlelés*** a felhasználók, illetve az alkalmazások elvárt viselkedésének modelljeit felhasználva az ettől való eltéréseket problémaként értelmezi. A rendellenesség-észlelési rendszerek fő előnye, hogy előzetesen tudják észlelni a támadásokat. Azzal, hogy meghatározzák, hogy mi a normális, ennek bármilyen megsértését azonosítani tudják függetlenül attól, hogy ez része-e a ***fenyegetési modell***nek (***threat model***) vagy nem. A módszer hátránya ugyanakkor a gyakori a téves riasztás és az, hogy az időben gyorsan változó rendszerekre nehéz az eljárást betanítani.
A ***visszaélés-észlelési*** rendszerek lényegében azt határozzák meg, hogy mi a rossz. Támadás leírásokat, más néven ***aláírás***okat (***signature***) tartalmaznak, melyeket összemérnek a felülvizsgálás adatfolyamával, keresve az ismert támadások tanújelét. A visszaélés-észlelési rendszerek előnye, hogy a felülvizsgálati adatok elemzésére irányul és ritkán vezet téves észleléshez. Ugyanakkor a módszer hátránya, hogy csak az ismert támadásokat tudja észlelni, amelyeknek van egy meghatározott aláírása. Ha egy új támadást felderítenek, azt a fejlesztőknek modellezni kell és hozzáadni az aláírás-adatbázishoz.
Fontos megemlíteni, hogy nem minden támadás jár együtt önműködően észlelhető, arra utaló jelekkel, egy rendszer feljogosított felhasználója általi visszaélés például nagyon nehezen észlelhető. Mégis a betolakodás-észlelés első módja a felhasználói tevékenység megfigyelése volt. Ekkor a szokatlan viselkedésekre figyeltek fel. Ilyen például ha egy felhasználó szabadságon van, s mégis helyileg be van jelentkezve. Az ilyen észlelés hátránya, hogy alkalmi jellegű és nem méretezhető bonyolult rendszerekre.
A betolakodás-észlelés fejlődése terén a következő lépés a operációs rendszer naplófájljainak figyelése, főként ***UNIX*** alapú rendszereknél. Több biztonsági segédprogram is erre épül, köztük az ismert ***Swatch*** (***Simple WATCHer for logfiles***), amelyről már esett szó a ***Linuxvilág 2001. augusztusi*** számában is. Az alkalmazásnak rendszernapló fájlok neveit adhatjuk meg, illetve azokhoz tartozó sorokat, részleteket. Ha az adott fájlban hibaüzenetnek megfelelő szövegrész jelenik meg (például ***file not found***), a ***Swatch*** előre meghatározott műveletet hajt végre: adott programot indít el, e-mailben értesíti a rendszergazdát stb.
Vállalati méretű rendszerek védelemre alkalmasak az összetett szerkezetű, ***hálózati betolakodás-észlelő rendszerek*** (***Network Intrusion Detection System, NIDS***). Ezek közül a kereskedelemben kapható ***RealSecure***, míg a ***Snort*** nyílt forráskódú. A ***Snort*** egy szabályleíró nyelvre épül, amely az adatminták, a hálózati protokollok és a rendellenességek vizsgálatát, illetve ezek kezelését is támogatja. Elsősorban a hálózati forgalom megfigyelésére (szonda) alkalmas, jól konfigurálható rendszert valósít meg; a szabályok (aláírások) adatbázisát az Internetről folyamatosan tudja frissíteni. A ***Snort*** fejlesztői és a felhasználói közössége által létrehozott új szabályok így azonnal beépülhetnek a szoftver által használt adatbázisba.

Válasz a betolakodásra

Egy probléma észlelésekor a betolakodás-észlelő rendszer által végrehajtott tevékenység sokféle alakot ölthet.
A legközönségesebb egy riasztás létrehozása, amely leírja az észlelt betolakodást. De a válasz lehet támadóbb is, mint amilyen egy rendszer adminisztrátor felhívása, egy sziréna megszólaltatása vagy éppen egy ellentámadás beindítása.
Az ellentámadás magába foglalhatja egy útválasztó újraalakítását úgy, hogy zárolja a támadó címét vagy még meg is támadja a tettest. Természetesen a támadó válasz veszélyes is lehet, mivel lehet, hogy ártatlan áldozat ellen indul. Erre példa az, hogy ha egy támadó a hálózatot ***átejtéses forgalom***mal (***spoofed traffic***) terheli meg. Az ilyen forgalom egy adott címről származóként jelenik meg, de ténylegesen máshol hozták létre. Ha a betolakodás-észlelő rendszer a támadás észlelése után újraalakítja a hálózati útválasztókat úgy, hogy zárolják a tettetett címről érkező forgalmat, akkor ez hatásában egy ***szolgáltatásmegtagadásos*** (***Denial of Service, DoS***) támadás végrehajtása lesz a vétlen hellyel szemben.

A Komondor részei

Különböző kiszolgálókon a programnak egyforma példányai futnak és figyelik a hálózaton észlelt esetleges betolakodási kísérleteket.



2. ábra Szabályozott elárasztás a Gnutella átfedőben

Ha egy ***Komondor*** egyed felismer egy, az általa felügyelt rendszer ellen irányuló támadást, akkor

- az általuk felügyelt rendszer védelmét erősíti, ami jellegzetesen a tűzfal szigorítását jelenti;
- a hálózaton keresztül értesíti a többi szoftveregyedet.

Az egyedek így egymást védelmezik. A védelem szempontjából lényegtelen, hogy egy adott támadó rosszindulatú ténykedését melyik egyed érzékelte először. Ha a támadási kísérletet egyszer rögzítette valamelyik egyed, akkor a többi a tapasztalat megosztása révén már felkészülhet (lásd az *1. ábrát*). A rendszer feladatai két részre oszthatóak, helyi és hálózati feladatkörre.

Tapasztalatok szerzése

A ***Komondor*** jelenleg működő, ***Linux*** rendszeren működő változata a ***rendszernapló fájlok vizsgálatával*** gyűjt tapasztalatot. Ez tulajdonképpen bármilyen betolakodási forma felderítésére alkalmas lehet, ugyanis a fent említett programok (például ***Snort***) is képesek ilyet vezetni. A naplókban többféle hibaüzenet jelenhet meg, amelyek támadásra utalnak.

Ilyen lehet bejelentkezési kísérlet nem létező felhasználói néven, vagy több egymás utáni kísérlet nem létező fájl letöltésére a ***HTTP*** kiszolgálón. Íme például egy tetten ért ***SSH*** féreg:

```
Sep 30 05:19:46 horka
➥sshd[6244]: Failed password
➥for illegal user munka from
➥::ffff:192.188.242.112 port
➥1391 ssh2
Sep 30 05:19:46 horka
➥sshd[6246]: Illegal user
➥munkaugy from
➥::ffff:192.188.242.112
Sep 30 05:19:46 horka
➥sshd[6250]: Illegal
➥user juhasz from
➥::ffff:192.188.242.112
```

A ***Komondor*** több naplófájlt is képes egyidejűleg követni. Az egyes naplófájlokhoz különböző kereső karakterláncokat adhatunk meg, amelyek a fentiekhez hasonló hibákra utalnak. Az egyes karakterláncok tetszőleges hosszúságú úgynevezett ***szabályos kifejezések*** (***regular expression***) lehetnek, amelyekhez két további idő paraméter is tartozik. Egyik, hogy az adott karaketrlánc milyen gyakori ismétlődése jelent támadást, ez másodperc nagyságrendű. (Például adott címről három másodpercen belül két bejelentkezési kísérlet nem lehet egy igazi felhasználó, aki csak a jelszavát elgépelte.) Másik, hogy a letiltás, a ***Komondor*** által a támadóval szemben életbe léptetett biztonsági szabály mennyi ideig legyen érvényes. Bármilyen megkötésről van is szó, nem lehet örök érvényű. Például ha a tiltás alapjául a támadó *IP* számát vesszük, annak a tulajdonosa változhat dinamikus *IP* szám kiosztás esetén. Ez inkább óra, vagy nap nagyságrendű időintervallum, bár támadó pásztázás esetén egy néhány perces blokkolás már eredményt hoz.

A Komondor által nyújtott védelem

Felmerült a kérdés, hogy a ***Komondor*** hol avatkozzon be a saját gazdagépe működésébe, a betolakodások elkerülése végett. Alkalmazási szinten történő beavatkozáshoz az egyes szolgáltatást nyújtó programok módosítására lehetne szükség. A szolgálat forráscím alapján történő megtagadását ugyanis nem mindegyik alkalmazás támogatja. Gondot jelent az is, hogy nagyobb méretű programok (például az ***Apache***) lassan indulnak, és a módosított beállításaikat is lassan olvassák így be. Az első változat ezért inkább a tűzfalon keresztül védi meg a rendszert, eldobásra kijelölve a támadó *IP* számról érkező csomagokat. A statikus csomagszűrés ezután a megadott ideig a műveleti rendszer feladata. Az idő leteltével a ***Komondor*** törli a tűzfalból a tiltó szabályt. A program így egy olyan speciális ***adaptív tűzfalként*** viselkedik, amely a tapasztalatait az alkalmazási szinten, illetve az alkalmazási szint felett gyűjti. A ***Linux*** rendszerben az ***iptables*** paranccsal hangolhatjuk a rendszermagba épített tűzfalat, amely egy statikus csomagszűrő, a hangolását szoftverünk biztosítja.

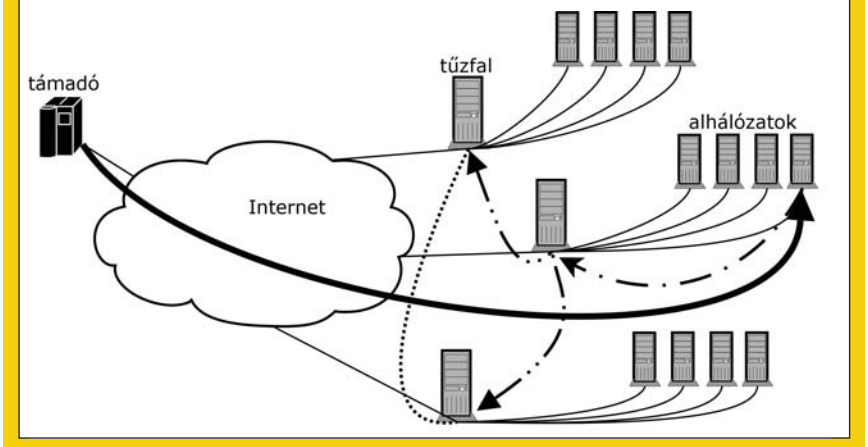
A hálózati modell kiválasztása

A szerzett tapasztalatokról a rendszer adatbázist vezet, és a többi egyeddel megosztja azokat. Az adatbázis megosztásának sebessége nagyban függ az alkalmazott hálózat felépítésétől. A választás két szempont miatt is a ***P2P*** topológiára esett. Egyrészt ez a tapasztalatok gyorsabb megosztását teszi lehetővé nagyobb számú csomópont esetén. Másrészt, egy ilyen hálózat jobban eltűri a csomópontok

megbízhatatlanságát, míg egy központosított (ügyfél / kiszolgáló) felépítésű rendszer a központ kiesése esetén azonnal működésképtelenné válik.
A megosztás gyors és megbízható kivitelezéséhez a *P2P* hálózatok közül is a *Gnutella* alkalmazásban is használt felépítés tűnt a legalkalmasabbnak.
A *Gnutella* a csomópontokat *serventnek* nevezi, mivel azok kiszolgálók és ügyfelek is egyben (*server+client*). Az új csomópont a hálózatba belépéskor egy úgynevezett *horgonyhoz* (*anchor*) kapcsolódik, amelytől megkapja az éppen működő csomópontok hálózati címét. Ezek után megpróbál kapcsolódni ezekhez az egyenrangúakhoz egy *ping* üzenettel; a kapcsolódási pontokat kínáló csomópontok *pong* üzenettel válaszolnak neki. A keresés úgy történik, hogy az egyenrangú szabályozottan elárasztja a kérdezését hét csomópont felé (2. *ábra*). Az ilyen kérést fogadó csomópont, függetlenül attól, hogy ő maga talált-e egyezést, ugyancsak hét másik csomópontnak küldi tovább. A mi programunkban a tapasztalatok továbbítása történik ehhez hasonló módon.
Az egyenrangúak legalább három, de legfeljebb öt másik egyeddel tartanak kapcsolatot. Ezek a paraméterek a programban egyszerűen beállíthatóak, segítségükkel vezérelhető a hálózat sűrűsége. A sűrűbb hálózat gyorsabb tapasztalatcserét és nagyobb megbízhatóságot biztosít, de nagyobb forgalmat is eredményez, ugyanis az egyedek nem fa-gráfba rendeződnek, így egy tapasztalat adatai több úton is érkezhetnek.
A rendszer indítása után az egyedek nem várják meg, hogy találjanak három megfelelő csatlakozási pontot az átfedőhöz. A rendszernapló fájlok vizsgálatát minél hamarabb el kell kezdeni, hogy a támadások észlelése elkezdődhessen. Ha az egyednek még nem sikerült csatlakoznia az átfedő egyetlen tagjához sem, a tapasztalatait akkor is rögzíti az adatbázisában. Legalább egy kapcsolat felépülése után a tapasztalatok már közvetíthetőek a többi csomópont felé.

A Komondor hatékonysága

A *Komondor* rendszer *hatékonysága* elsődlegesen az azt felépítő egyedek változatosságán múlik. A biztonsági



3. ábra Alhálózatok védelme Komondor hangolt tűzfalakkal

réseket kihasználó támadások szoftver-, illetve verzióspecifikusak.
A különböző szolgáltatásokat nyújtó hálózati programok (például *sshd*, web kiszolgáló) más-más verziói, fajtái lehetnek telepítve az egyes egyedeken. Az észlelt és remélhetőleg kivédett támadás után a *Komondor* segítségével a sebezhető kiszolgálók is megvédhetőek lehetnek. Ez három gyakran előforduló helyzetben következhet be:

- a kiszolgálók ugyanannak a szoftvernek különböző változatait futtatják (például *Apache 2.0.54* és *Apache 2.0.30*).
- a kiszolgálók ugyanazt a szolgáltatást különböző programokkal valósítják meg (webkiszolgálóként használható például az *Apache* vagy a *Zeus* is);
- a kiszolgálók más műveleti rendszerekre épülnek. (például *Linux* és *Windows*)

Az első esetben, amikor a kiszolgálók ugyanannak a programnak eltérő változatait futtatják, a sebezhetőségi pontok az egyes gazdagépeknél hasonlóak lehetnek, ugyanis a hibák azok felderítéséig a programokban rendszerint verzióról verzióra öröklődnek. A második és harmadik esetben viszont az egyes szolgáltatások sebezhetőségi pontjai általában diszjunktak, ami miatt az egyes biztonsági rések elfedése hatékonyabb lehet.
A *Komondor* hatásfokát korlátozza az *IP* címek nagy száma. A jelenleg használt protokoll, az *IPV4* esetén a címtartomány 2^{32} méretű, *IPV6* esetén ennél is jóval nagyobb (128 bites hálózati címek). Annak valószínűsége, hogy egymástól távol lévő *IP* számú gazdagépeket rövid időn belül ugyanaz a támadó vesz célba, rendkívül alacsony. Ahogyan korábban már említésre került, gyakori viszont az egymáshoz közeli címek rövid időn belül, sorrendben történő vizsgálata, amikor a támadó egy adott tartományt pásztáz végig célpontok után kutatva. Érdemes emiatt a *Komondort* használó rendszergazdának úgy szerveznie a hálózatot, hogy a tartomány első gazdagépe, a fenti példában a *194.143.224.1* című, különösen biztonságos legyen (kevés, szokatlan azonosítóval rendelkező felhasználó, bonyolult jelszavak, kevés szolgáltatás stb.). A tartomány támadó célú átvizsgálása ennél a gépnél kezdődik, és a védelem így a lehető leghamarabb reagálhat. A támadó az alhálózati tartomány átvizsgálásakor kétféleképpen járhat el:

- Felderítheti, hogy mely gépeken fut az általa támadni kívánt szolgáltatás, utána próbálva kihasználni a hibákat.
- Fordítva, a hibák vizsgálatát (például a belépési kísérlet gyenge jelszavakkal) az *IP* számok szerint haladva sorban elvégzi.

Mind a két támadói magatartás eseté egy *Komondor* egyednek valós esélye van a támadó hálózati címének „szétkürtölésére”, mivel hasonló *IP*

címek esetén valószínűleg helyi hálózaton dolgozik a többi *Komondor* egyeddel, vagyis közöttük az összeköttetés fizikai szinten gyorsabb, mint a távoli támadó és az egyes helyi hálózatbeli gazdagépek között.

Biztonság és robusztusság

A program legnagyobb hátránya, hogy az átfedőbe sajnos könnyen rosszakaratú egyed épülhet. Egy ilyen a legnagyobb kárt hamis tapasztalatok megosztásával tud okozni, ezek ugyanis *szolgálatmegtagadás*t (*DoS*) okoznak. A biztonság és hatékonyság között a megszokott módon csak kompromisszum lehetséges. Megoldás lehet ez ellen az átfedőbe csatlakozó egyedek felhasználói jogainak vizsgálata (*ident vizsgálat*). Ehhez felhasználhatjuk azt a tényt, hogy a *Komondor* csak rendszergazda jogosultságokkal futhat az egyes egyedek gazdagépein, különben nem tudna naplófájlokat vizsgálni és a hálózatba beavatkozni. Vagyis felhasználói jogosultságokkal rendelkező rosszakaratú egyén nem tudná futtatni a *Komondor* szoftvert. Ha azonban egy *Komondor* mégis felhasználói jogokkal fut egy gazdagépen, akkor az valószínűleg hamis tapasztalatokat hirdet, vagy más módon zavarja a rendszer integritását. Sajnos az *ident* szolgáltatás általában csak *UNIX* típusú rendszereken ismert. *Windows* platformon rendszerint meghamisított *ident* kiszolgáló fut valamilyen külső elvárás miatt.

A rendszer kritikus pontja a horgonypont. Ennek kiesése esetén új egyed nem képes kapcsolódni az átfedőhöz. A már kapcsolódott egyedek között a tapasztalatcsere megmarad a *P2P* elvnek köszönhetően.

A továbbfejlesztés irányai

Ebben a fejezetben kerül bemutatásra a *Komondor* rendszerének lehetséges továbbfejlesztési irányai, melyeket az előbbiekben alkalmazott szempontok szerint értékelünk.

Az *alhálózatok védelméhez* megfontolandó a *Komondor* átfedő hálózatának részben központosított kialakítása, hasonlóan a *FastTrack* hálózati modellhez. A *FastTrack* eredeti célja ugyan a méretezhetőség javítása volt, azonban a következőkben ismertetésre kerülő megfontolások alapján szintén a részben központosított átfedő látszik célszerűnek. Tudjuk, hogy hálózatok, géptermek kialakításakor megszokott módszer, hogy az egyes hálózati szelvények munkaállomásaihoz közös tűzfal tartozik. Mivel ilyenkor a munkaállomások általában erre támaszkodnak, önmaguk nem látnak el tűzfal feladatkört. A védelmet nyújtó *Komondor*nak mindenképpen a tűzfal számítógépén kell futnia. A munkaállomásokon futó *Komondor* egyedeknek a feladata ekkor a betolakodás-észlelésre korlátozódik. A megszerzett tapasztalatokat, amelyekről adatbázist sem szükséges vezetniük, azokat mindegyik a saját tűzfalán futó felsőrangújához továbbítja. Természetesen a felsőrangúak is észlelhetnek betolakodási kísérletet.

A *P2P* elv ebben az esetben ugyanúgy alkalmazható, mint eddig. A tűzfalak között jön létre ekkor a *P2P* összeköttetés, amely lehet teljes gráf is, azaz mindegyik mindegyikhez, mivel a forgalom még egy adott tapasztalat száz másik gép felé történő megosztásakor is kilobájt nagyságrendű csupán. A már ismertetett címtartomány pásztázás miatt pedig itt szomszédos alhálózatokat védhetnénk hatásosan, ami úgyszintén kis számú felsőrangút feltételez.

Ez a fajta védelem igen hatásos lehet, azonban a hálózati/szállítási réteg közötti beavatkozásra korlátozódik. A munkaállomások *Komondor* egyedei egy egyszerűbb programot futtatnak, mint a felsőrangúak. A *P2P* hálózatba szerveződés helyett ezek a hálózati átjáró (gateway) címét kérdeznék le az operációs rendszertől, amely értelemszerűen megegyezik a tűzfal címével, ahol a felsőrangú egyed fut. Az építménynek akkor is van értelme, ha az egyes tűzfal mögötti gazdagépek *közös hálózati címen osztoznak*.

Ekkor a *hálózati címfordítás*t (*Network Address Translation, NAT*) végző útválasztó a kapcsolat kapuszáma szerint más-más belső ponthoz továbbít csomagokat. Ezek a csomagok eredetileg hozzá érkeznek, de a szolgáltatásokat érő támadásokat csak az azokat megvalósító gazdagépek képesek felderíteni. Ennek az oka például az, hogy az útválasztónak nincs arról tudomása, hogy egy adott felhasználói név egy belső gépen létezik-e.

A betolakodás-észlelés további módszerei

A *Komondor* megvalósított változata bővíthető további tapasztalatszerző módszerekkel. Betolakodási kísérletek észleléséhez lehetséges például egy olyan program modul létrehozása, amely egy web *kiszolgálót imitál*, azaz tényleges szolgáltatásokat nem nyújt, csak a bejövő *HTTP* kérések vizsgálatával próbál gyanús jeleket gyűjteni. A támadó számára megtévesztő lehet, ha az igazi web kiszolgálót egy szokatlan hálózati kapura (port) telepítjük, például a 8080-asra, míg a szokásos 80-as kapun a *Komondor* imitált web kiszolgálója fut. Egy későbbi windowsos változatban erre a célra felhasználható lehetne az ott egyébként nem használt 22-es (*SSH*) vagy 23-as (*telnet*) kapu. Ezt a modult nem szükséges közvetlenül a *Komondor*ba építeni, futhat külön folyamatként is, naplófájlt vezetve.

Az előbbiekben ismertetett saját tapasztalatszerzési eljárás mellett számítani lehet arra, hogy a *Snort* közösség egy mindig naprakész felderítő adatbázissal rendelkezik, így a fejlesztés során a hangsúlyt a saját tapasztalatszerzés helyett a *Komondor* egyedek P2P hálózatba szerveződésének javítására érdemes fektetni.

Czirkos Zoltán

Jelenleg diplomatervező a Budapesti Műszaki Egyetem Elektronikus Eszközök Tanszékén. Kutatási területe az operációs rendszerek betörésvédelme és a P2P kommunikáció. 2005-ben a Tudományos Diákköri Konferencián II. helyezést ért el. Kedvencei a boszorkányos és a rózsaszín párducos filmek.

KAPCSOLÓDÓ CÍMEK

Komondor tesztváltozat:
➲ http://jutas.eet.bme.hu/

Snort:
➲ http://www.snort.org

RealSecure:
➲ http://www.iss.net

Swatch:
➲ http://swatch.sourceforge.net

Nmap:
➲ http://www.insecure.org

Gnutella:
➲ http://www.gnutella.org