Alkalmazásszintű proxyzás a Zorppal (1. rész)

A különféle támadási formák ellen az alkalmazásszintű proxyk nyújtják a legjobb védelmet, viszont bonyolultabbak is a csomagszűrőknél.

Első ránézésre úgy tűnik, hogy az állapotalapú csomagszűrés teljesen uralma alá vonta a tűzfalak világát, nemcsak a piaci részesedés, de gondolkodásmódunk tekintetében is. Az állapotalapú csomagszűrést végző termékek listája meglehetősen hosszúra nyúlik, a kereskedelmi fejlesztésű Check Point Firewall-1 és a Linux kiváló Netfilter magkódja egyaránt ide sorolható.
De vajon mi a helyzet az alkalmazásszintű proxykkal? A tűzfalakkal foglalkozó mérnökök régóta bizonygatják, hogy semmi sem képes olyan sokféle hálózati támadást megállítani, mint az alkalmazásokat pontosan ismerő proxy. ***Scheidler Balázs***, a ma már nélkülözhetetlennek számító naplózóprogram, a `syslog-ng` készítője, megalkotta a Zorpot, ami egy nyílt forrású, egész egyszerűen zseniális proxytűzfal. Ebben a hónapban azt szeretném elmesélni, hogy az alkalmazásszintű proxytűzfalak kérdésében micsoda megváltásnak éreztem a Zorp megjelenését, és e program hogyan lehet mindazok segítségére, akiknek fontos és érzékeny adatokat kezelő hálózatok biztonsága felett kell őrködniük.

Frissítsük fel emlékezetünket

Talán többen is feltették már a kérdést, mi a csuda az az alkalmazásszintű proxyzás és az állapotalapú vizsgálat? Miért jobb az egyik vagy a másik?
A tűzfal egy számítógép vagy egy beágyazott programot futtató eszköz, amely képes a hálózatok egymástól való elkülönítésére és a közöttük folyó forgalom szabályozására. Azokat a szabályokat, amelyek megmondják, mely hálózati csomópontok milyen típusú hálózati csomagokat és hová továbbíthatnak, tűzfalszabályoknak, összességüket pedig tűzfal-házirendnek nevezzük.
Ezek a szabályok különböztetik meg a tűzfalakat a normál útválasztóktól. Az útválasztókon a csomagok különféle hálózatok közötti mozgatására vonatkozó szabályokat nyilván meg kell adni, de azt már nem feltétlenül kell előírni, hogy milyen forgalmat hova szabad továbbítani. Ezzel szemben a tűzfal képes a kifinomult válogatásra.
A csomagok szétválogatásának egyik legegyszerűbb szempontja az internetprotokoll (IP) fejrészének tartalma. Az IP-fejrész alapszintű adatokat tartalmaz, a legfontosabbak a protokolltípus, a forrás és a célállomás címe, illetve – ha van ilyen – a forrás- és a célkapu. A kapuszámokat valójában a csomag következő, az UDP vagy a TCP protokollhoz tartozó fejrésze tartalmazza. Azokat a tűzfalakat, amelyek csak az alapszintű adatokat vizsgálják, egyszerű csomagszűrőknek nevezzük. Mivel az egyszerű csomagszűrők nem néznek bele a csomagok mélyébe, működésük nagyon gyors. Ugyanakkor az IP-fejrész és a TCP- vagy az UDP-kapuszám semmit nem árul el a csomag más csomagokkal fennálló viszonyáról. Ha például megvizsgáljuk egy HTTP-csomag IP-fejrészét, akkor az IP-fejrészbeli protokollmezőnek köszönhetően tudni fogjuk, hogy TCP-csomagról van-e szó, a forrás- és a cél-IP-címéből meg tudjuk állapítani, honnan érkezik és hova küldték, a célkapu száma alapján pedig el tudjuk dönteni, milyen típusú alkalmazás küldte. Az *1. táblázat* egy egyszerű csomagszűrő egyik szabályát szemlélteti. Ilyen szintű vizsgálatnál azonban fontos, hogy a HTTP-kapcsolattal összefüggő adatokat figyelmen kívül hagyjuk. Nem nézzük, hogy a csomag új HTTP-kapcsolatot hoz-e létre vagy már meglévő kapcsolat csomagja, esetleg véletlenszerűen, adott esetben rossz szándékkal elküldött, más csomagokkal semmilyen viszonyban nem lévő csomagról van-e szó. Annak oka, hogy mindezt figyelmen kívül hagyjuk, hogy a kapcsolatra vonatkozó adatok, mint a TCP-zászlók (TCP flags), a TCP-sorozatszámok (TCP sequence numbers) és az alkalmazásszintű parancsok a csomag belsőbb részeiben helyezkednek el, ahova a csomagszűrő már nem lát be. Itt lép a képbe az állapotalapú csomagszűrés. Az állapotalapú csomagszűrők, ahogy az egyszerű csomagszűrők is, az egyes csomagok forrás- és cél-IP-címének, illetve forrás- és célkapujának vizsgálatával kezdik munkájukat. Ennél azonban mélyebbre is beásnak a csomagba, és annak UDP- vagy TCP-fejrésze alapján megvizsgálják, hogy a csomag tartalma egy új kapcsolat létrehozására irányul-e. Ha igen, a tűzfal létrehoz egy új bejegyzést az állapottáblában. Ha nem, akkor megvizsgálja az állapottábla tartalmát, és megállapítja, hogy a csomag már fennálló kapcsolathoz tartozik-e. Az állapotalapú csomagszűrő képes azoknak a csomagoknak a kiszűrésére, amelyek valótlanul állítják magukról, hogy meglévő kapcsolathoz tartoznak.
Itt kell megemlítenünk, igaz ugyan, hogy az UDP kapcsolat nélküli protokoll, ám egy jó állapotalapú tűzfal tudja, hogy például egy az 53-as UDP-kapura küldött, kimenő DNS-kérés hatására érkezni fog egy válasz a kiszolgáló 53-as kapujáról. Az állapotalapú csomagszűrés két fontos előnnyel bír az egyszerű csomagszűréshez képest.

1. táblázat *Egyszerű csomagszűrő szabályok a HTTP-forgalomhoz*

Forrás-IP (Source IP)	Cél-IP (Destination IP)	Protokoll	Forráskapu (Source Port)	Célkapu (Destination Port)	Művelet
Bármely	192.168.1.1	TCP	Bármely	80	Engedélyezés
192.168.1.1	Bármely	TCP	80	Bármely	Engedélyezés

2. táblázat *Állapotalapú csomagszűrő szabály a HTTP-forgalomhoz*

Forrás-IP	Cél-IP	Protokoll	Forráskapu	Célkapu	Állapot	Művelet
Bármely	192.168.1.1	TCP	Bármely	80	Új	Engedélyezés

Először is a tűzfalszabályok egyszerűbbek lehetnek. Nincs szükség rá, hogy minden egyes kétirányú – például HTTP – kapcsolat mindkét irányát pontosan leírjuk, a tűzfalszabályokban csak a megengedett kapcsolatok kezdeményező pontját kell megadnunk. A megengedett és már létrehozott kapcsolatokhoz tartozó további csomagok kezelése a tűzfal állapottáblája alapján történik, az általunk megadott tűzfalszabályoktól függetlenül. A 2. *táblázat* alapján látható, hogy egyetlen szabállyal is engedélyezni tudtuk ugyanazt a HTTP-kapcsolatot, amelynek leírásához az 1. táblázatban még két szabályra volt szükségünk.

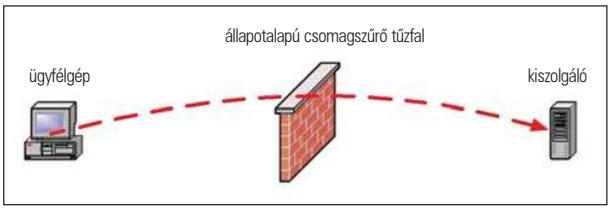
Az állapotalapú csomagszűrés második fontos előnye, hogy nem kényszerülünk olyan, a nem állapotalapú megoldásoknál időnként szükséges lépésekre, mint az 1024-nél magasabb célkapuszámmal rendelkező, bejövő TCP- és UDP-csomagok beengedésére az internetről a belső hálózatra. Ilyesmire akkor lehet szükség, ha nincs más lehetőségünk a megengedett kapcsolatok és a csomagok összerendelésére. Nem szaporítom tovább a szót, a lényeg az, hogy az állapotalapú csomagszűrés jobb védelmet nyújt, mint az egyszerű csomagszűrés.

„Nagyszerű – állapíthatjuk meg – az állapotalapú csomagszűrők tehát hatékonyabbak és biztonságosak!" – valóban így van. De vajon vannak-e olyan dolgok, amelyeket az állapotalapú csomagszűrők sem vesznek figyelembe? Mi történik például a hibásan megformált HTTP-kérésekkel és a szándékosan átlapolt IP-töredékekkel (overlapping IP fragments)? Létezik olyan tűzfal, amely minden egyes csomagot teljes egészében képes megvizsgálni, vagy valamilyen más módon tud segíteni abban, hogy a lehető legkevesebb ártó szándékú csomag érje el a hálózatunkat? Igen, létezik ilyen, ezt nevezzük alkalmazásszintű proxynak, más szóval alkalmazásrétegbeli átjárónak. Míg a csomagszűrők – egyaránt állapotalapúak és egyszerűek – minden csomagot megvizsgálnak, majd az engedélyezetteket továbbítják, addig az alkalmazásszintű proxyk minden létrehozandó kapcsolatot két részre osztanak, önmagukat pedig az összeköttetés egyenrangú résztvevőjeként iktatják be középre. Az ügyfél, vagyis a kapcsolat kezdeményezője számára a tűzfal kiszolgálóként látszik, míg a megjelölt cél, vagyis a tényleges kiszolgáló számára ügyfélként jelenik meg. Az *1.* és a *2. ábra* a két megoldás közötti különbséget szemlélteti. Az 1. ábrán egy állapotalapú csomagszűrőt láthatunk, amely „csupán" engedélyezi és tiltja a kapcsolatokat. Amolyan megfigyelőként foghatjuk fel, amely az engedélyezett csomagokat többé-kevésbé változatlan formában továbbítja – kivéve azt az esetet, ha például hálózati címfordítást (Network Address Translation, NAT) használunk. Ezzel szemben a 2. ábrán azt láthatjuk, hogy a tűzfal a megengedett kapcsolatok végpontjaként önmagát jelöli meg, majd új, proxyzott kapcsolatot kezdeményez az eredetiek célállomásaival.

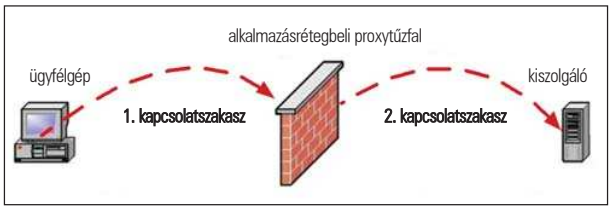
A proxyzás kétféle típusú lehet: átlátszó (transparent) és nem átlátszó. Átlátszó proxy használatakor egyik fél sem veszi észre, hogy kapcsolatuk valójában proxyn keresztül jött létre. Az ügyfélrendszer úgy címezi meg a csomagjait, mintha nem is lenne tűzfal a hálózatban, vagyis a valódi célállomás IP-címét adja meg. A nem átlátszó proxynál viszont az ügyfélnek a saját csomagjait a tűzfalnak szükséges címeznie, és nem a valódi célállomásnak. Mivel az ügyfélnek ilyenkor valamilyen módon közölnie kell a tűzfallal, hogy valójában kivel szeretne kapcsolatba lépni, nem átlátszó proxy használatakor proxyképes alkalmazásokat (proxy-aware applications) kell futtatnunk. A legtöbb webböngésző és FTP-ügyfél ugyan képes a nem átlátszó proxyk használatára, egy átlátszó proxy azonban kevesebb bosszúságot okoz a felhasználóknak. A korszerű alkalmazásszintű proxyk, mint a Zorp is, átlátszók.

Átlátszó vagy sem, a proxyzás számos előnnyel kecsegtet. Az egyik ilyen, hogy az alacsonyabb szinten hibás csomagokat, például a fejrészükben rendellenes jelzőkkel ellátott IP-csomagokat a tűzfal nem küldi tovább. A tűzfal a másodlagos kapcsolatot olyan módon hozza létre, amit ő maga tart helyesnek, és nem az ügyfélgép. Másik előnye abból fakad, hogy a tűzfal teljes egészében új kapcsolatokat hoz létre, nem csupán továbbküldi vagy átírja a hozzá beérkező csomagokat, és így kiváló helyzetben van ahhoz, hogy alkalmazásszinten vizsgálhassa meg a kapcsolatot. Ne gondoljuk, hogy ez magától értetődő képesség, ha például a tűzfal csak SOCKS tűzfal, és nem valódi alkalmazásrétegbeli proxy, akkor megteheti azt, hogy az ügyfelektől beérkező csomagok adatait egyszerűen átmásolja az új, proxyzott csomagokba. Ha a tűzfal képes az alkalmazásszintű működésre – ilyen például a Zorp –, akkor arra is alkalmas, hogy az ügyfelek által küldött csomagok tartalma alapján bizonyos döntéseket hozzon meg.

Nézzünk egy példát! Tegyük fel, hogy van egy nyilvános webkiszolgálónk, ezt eredményesen lehet támadni egy hibás formátumú `HTTP GET` paranccsal, amely például túlságosan hosszú URL-t tartalmaz, és ezzel átmenetitár-túlcsordulást



1. ábra Állapotalapú csomagszűrő használatakor a csomagok az ügyfél és a kiszolgáló között közvetlenül utaznak, feltéve, hogy továbbításuk valamelyik szabály vagy állapottábla-bejegyzés alapján megengedett



2. ábra Alkalmazásszintű proxy használatakor a kapcsolat két részre osztható: az ügyfélnek a tűzfal kiszolgálóként látszik, a kiszolgálónak pedig ügyfélként viselkedik

(buffer-overflow) idéz elő. Alkalmazásszintű proxytűzfalunk fogadja az ügyféltől érkező kérést, ám amikor látja, hogy túlságosan hosszú URL-t tartalmaz, akkor az ügyfélnek hibaüzenetet, a kiszolgálónak pedig `reset` parancsot küld, és a támadócsomag továbbítása nélkül megszakítja a kapcsolatot.
Az előnyök mellett azonban hátrányokkal is számolni kell. A proxyzás eleve több erőforrást köt le, mint a csomagszűrés, és az alkalmazástudatos proxyzásra ez még inkább igaz. Az alkalmazásszintű proxyknak ezt a hátrányát sokszor túlhangsúlyozzák. Például a Zorp egy 700 MHz-es Celeron processzorral és 128 MB memóriával felszerelt gépen 88 Mb/s sebességgel képes a HTTP-forgalom vizsgálatára, ami közel kétszerese egy T3 WAN-kapcsolat sebességének. Ha a Zorp alá egy kétprocesszoros Pentium III gépet, 512 MB memóriát és SCSI RAID merevlemezeket teszünk, akkor körülbelül 480 Mb/s sebességre lesz képes, legalábbis a ➲ http://www.balabit.hu címen elérhető Zorp Professional v2 leírása szerint.
Az alkalmazásszintű proxyk tehát képesek rá, hogy a hálózati kapcsolatokba beékelődve minden csomagot nulláról indítva újra létrehozzanak, illetve intelligens döntéseket hozzanak arról, mely alkalmazásszintű parancsokat és adatokat szabad továbbítani – vagyis magasabb szintű védelmet nyújtanak. Munkájukat az egyes alkalmazások feltételezett működésének ismeretében, és nem csupán a hálózati csomagok kinézete alapján végzik. A legfőbb ellenérv az alkalmazásrétegbeli proxyk ellen a teljesítményre vonatkozik, ám – hála Moore törvényének – az egyre olcsóbbá és nagyobb teljesítményűvé váló gépeknek köszönhetően ez is könnyedén elhárítható.
Természetesen nem szabad elmenni egy másik, az alkalmazásszintű proxyk kapcsán sokak által felvetett tényező mellett, azaz jóval összetettebbek. Az alkalmazásszintű proxyk értelemszerűen sokkal kifinomultabbak, mint a csomagszűrők, és sokkal bonyolultabb a beállításuk is, ami nem meglepő, hiszen egy sportautó vezetése is nagyobb tudást kíván, mint egy taxi leintése. Egy Zorpot futtató tűzfalat vagy egy Secure Computing Sidewinder készüléket beállítani mindig is nehezebb lesz, mint egy Check Point Firewall-1 vagy Linux Netfilter/IP Tables alapú megoldást.
Vajon a nagyobb fokú biztonság nem éri-e meg azt a kis többletmunkát? Mint máskor is, mindenki maga döntse el, hogy mivel éri meg foglalkoznia. Lesz, aki úgy gondolja, érdemes dolgoznia a magasabb szintű védelemért, és lesz, aki nem öl bele több energiát. Bármi is legyen a végső döntés, remélem, meghozásához sikerült némi segítséget adnom. Írásom fennmaradó részében röviden bemutatom a Zorp telepítését és beállítását.

A Zorp változatai

A Zorp proxydémon a Linux-rendszermag felett, a normál Netfilterrel és a Balabit Kft. által készített Tproxy rendszermag-modulokkal párhuzamosan fut. A Zorpot Debian Linuxon fejlesztik, és a hozzá mellékelt leírás túlnyomó részében is feltételezik, hogy a használó Debiant futtat.
A Zorp három változatban érhető el: a Zorp GPL az ingyenes, GPL szerződéssel elérhető változat; a Zorp Unofficial a Zorp GPL újdonságokkal bővített próbaváltozata; a Zorp Professional, vagy csak egyszerűen Zorp Pro pedig kereskedelmi termék, amely számos fontos szolgáltatást nyújt.
Ha megvásároljuk a Zorp Prót, akkor egy rendszerindításra alkalmas CD-ROM-ot kapunk, amely nemcsak a Zorp Prót, de az ZorpOS-t is telepíti. Utóbbi egy lecsupaszított, a Zorp igényeihez igazított Debian-változat. A Zorp Pro egy teljesen szűz gépre negyed óra alatt felrakható, természetesen a beállítások megadása ebben nincs benne.
A Zorp Pro az új Zorp Management Servert (ZMS) is tartalmazza, amelynek segítségével több Zorp tűzfalat is kezelhetünk egyetlen központi felügyeleti állomásról. Az állomás távoli működtetésére a ZMC szolgál, ez egy grafikus felületet kínáló ügyfélprogram, amely Debian Linux és Windows alá egyaránt elérhető. A ZMS szolgáltatásait tekintve egyenértékű a Check Point Firewall-1 felügyeleti moduljával, amely a legfontosabb tényező volt abban, hogy a Check Point meghódíthatta a vállalati tűzfalak piacát. A ZMS-nek köszönhetően a Zorp kiváló megoldást jelent a nagy számú tűzfallal rendelkező nagyvállalatok számára is.
A Zorp Próval ellentétben a Zorp GPL és a Zorp Unofficial telepítéséhez már működő Linux-telepítésre van szükség, amely tartalmazza a következő összetevőket: glib 2.0, Python 2.1, libcap 1.10 és OpenSSL 0.9.6g. Működéséhez IP, tűzfal és átlátszó (transparent) proxytámogatással fordított 2.2-es, vagy `iptables`, `iptables/netfilter` kapcsolatkövetés (conntrack), `iptables` NAT és `iptables/netfilter tproxy`-támogatással – ez a Balabit Tproxy rendszermag-foltjával érhető el (➲ http://www.balabit.com/products/oss/tproxy) – fordított 2.4-es rendszermag szükséges. Az említett szolgáltatások mindegyikét modulba kell fordítani. Ha az operációs rendszer készen áll, a Zorp GPL-t bináris `deb` csomagokból tudjuk telepíteni, de a forrásból való fordítást is választhatjuk.
A következő alkalommal arról lesz szó, hogyan kell beállítani a Zorp GPL-t egy átlagos internet – szabad zóna (DeMilitarized Zone) – megbízható hálózati topológia (Trusted Network Topology) védelmére.

Mick Bauer

Linux Journal 2004. március 119. szám