Tartalomszűrő tűzfalak teljesítményvizsgálata

Készítette: Adamkó Péter András

Konzulensek:

Szigeti Szabolcs - BME Informatika Központ

Major Csaba – BalaBit IT Security Kft.

Alulírott Adamkó Péter András, a Budapesti Műszaki és Gazdaságtudományi Egyetem hallgatója kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, és a diplomatervben csak a megadott forrásokat használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen a forrás megadásával megjelöltem.

Absztrakt

Az elmúlt években a hálózatok összekapcsolódásának következményeként a tűzfal, mint a hálózat kritikus eleme jelent meg, s mára mindenhol jelen van. Azonban amellett, hogy a közvetett és közvetlen támadások ellen védelmi pontként szolgál, új képességekkel is felruházható, mellyel kikényszeríthetjük a szervezet biztonságpolitikai szabályainak betartását. Erre leginkább alkalmas a tűzfalak legújabb generációja, a tartalomszűrő komponenssel rendelkező megoldások egyre szélesebb köre.

Dolgozatom első részében a tűzfalak fejlődési irányzatait veszem sorba, majd a tartalomszűrési megoldásokról adok számot.

Ezt követően a Zorp tűzfallal és annak konkrét tartalomszűrési lehetőségeivel fogok foglalkozni.

Legvégül pedig méréseket végzek a tartalomszűrés, protokollelemzés, csomagszűrés teljesítményigényeinek összehasonlítására.

Abstract

As a result of the converging networks, the firewall became a critical component in the past few years, and it is present in most network. Besides that it defends the network from directed and undirected attacks, it has acquired new features, and with the help of these features it can enforce security policies introduced in the organization. The most suiteble for this work is the continuously extending market of the next generation firewalls, which have content filtering component.

In the first part of my thesis I will cover the evolution of firewalls and the content filtering solutions.

The next part will cover the Zorp firewall and its special features, including content filtering.

In the last part I will measure the performance demands of content filtering, protocol analysis and packet filtering.

1. Bevezetés

Az elmúlt évek dinamikus fejlődése a hálózatok soha nem látott mértékű összekapcsolódását, integrációját hozta magával. Ez legfőképpen az Internet elterjedésének köszönhető, melynek egyenes következménye a gyors adat és információcsere iránti igény kialakulása. Azonban nem szabad elfeledkezni arról, hogy az Internet mindig is játszótere volt a felhasználóknak, s igen sok rosszindulatú, vagy csak „kíváncsi" található közöttük. Nem véletlen, hogy a számítógépes vírusok története is több mint 20 évre nyúlik vissza. Ez azonban csak az egyik veszély, amely az összekapcsolt hálózatokból ered. A betörésekből, adatlopásokból fakadó károk jelentős veszteségeket okoznak a legtöbb cégnek, a CSI/FBI 2005-ös felmérése alapján az első három helyen a vírusok, jogosulatlan hozzáférések, valamint védett információk lopása áll.

Ezen események vezették rá az embereket, hogy védelemre van szükség ugyanúgy, ahogy a valós világban is léteznek vagyonvédelmi rendszerek, eszközök. A számítógépes világban is egyre több ilyen áll rendelkezésünkre, vírusírtók, viselkedésdetektáló szoftverek, de talán a legrégebbi ilyen hálózati eszközünk a tűzfal, mely mára az elsődleges határvédelmi eszközzé vált, s egyre több funkciót építenek bele.

A tartalomszűrés egyike azon tulajdonságoknak, melyekkel a legújabb generációs tűzfalak rendelkeznek. Megfelelő szabályrendszerekkel kiszűrhető sok támadási (vírusok, puffer túlcsordulás, stb.) mód. Kétségtelenül logikus megoldás mindjárt a hálózat legszélén szűrni a forgalmat, azonban minél több funkciót valósít meg az ott levő eszköz, annál nagyobb erőforrásigénnyel rendelkezik, annál sérülékenyebb egy DoS (Denial of Service – túlterheléses támadás) támadásra. Viszont a folytonos és korrekt működés elengedhetetlen, ezért érdemes információkkal rendelkezni a tűzfal teljesítményéről, az esetleges funkcióvesztéssel, működési rendellenességekkel járó határértékekről.

Diplomatervem célja egy, a legújabb tűzfal technológiát magában ötvöző tűzfal teljesítménymutatóit megállapítani, protokoll analízis tartalomszűrés és csomagszűrés használatának esetén. Ehhez összeállítok egy, a mérésre alkalmas módszert, s a hozzá szükséges tesztrendszert, legvégül pedig értelmezem az eredményül kapott adatokat.

A dolgozat első részében a napjainkig elért eredményeket, s a rendelkezésre álló tűzfal technológiákat összegzem azok előnyeivel és hátrányaival. Itt kitérek a tűzfal topológiák jellemzőire, valamint a tartalomszűrési technológiákra. Ezen technológiák némelyike túlmutat a tartalomszűrésen, s a „totális kontroll” lehetőségét is felveti, mely több szempontból is aggályos lehet.

A második részben a felhasznált tűzfal szoftver részletesebb leírása található. Itt elsősorban a mérési feladathoz szükséges komponens leírásokat találhatjuk meg, azok finomhangolási lehetőségeivel. Ezt követően a különböző mérési módszerek alapján az általam fontosnak vélt mérőszámok felsorolása és magyarázata következik. A legtöbb ilyen mérőszám inkább alacsonyabb szintű adatokkal szolgál, s bármelyik tűzfal tesztelhető vele. Így aztán szükségessé vált alkalmazás rétegbeli protokollméréseket is futtatni, melyek HTTP objektumokat kezelnek a bit vagy csomag léptékű adatok helyett.

Legvégül sor kerül a tesztrendszer megtervezésére, illetve más rendszerekkel való összehasonlítására, hiszen eszközök már rendelkezésre állnak, de komoly hátrányokkal rendelkeznek. Ilyen például konfigurálhatóságuk, kevesebb teszteset fedhető le velük, valamint ún. ”black box”-ként működnek, nem látunk bele működésükbe. A megtervezett tesztrendszer egy kisebb hálózati szegmenst szimulál, munkaállomásokkal, szerverrel, tűzfallal és a tartalomszűrésért felelős hosztokkal. Mivel véges számú erőforrás áll rendelkezésre a megvalósítás során, ezért külön teljesítménytesztelő programok felhasználása szükséges a mérés során, melyek képesek maximális terhelést adni a hálózatra és a tűzfalkomponensekre. Többfajta tűzfal topológia is mérésre kerül, mely plusz adatokat adhat egy valós rendszer skálázására. Végezetül sor kerül a konkrét mérések leírására, azok eredményeivel, s az eredményekből levonható következtetésekkel. A mérések leírása során külön figyelmet fordítok arra, hogy a tesztsorozat bármikor rekonstruálható legyen.

2. Tűzfalak

2.1. Tűzfalak történelme

Az Internet kialakulásakor csak egy viszonylag kis felhasználói közösséggel rendelkezett, mivel ekkor még igencsak korlátozottan volt elérhető az emberek számára. Főként az akadémiai hálózatok felhasználói részesülhettek a hálózatok összekapcsolásából származó szabad információáramlás hasznából. A kutatók a nyíltságot és az együttműködési lehetőségeket tartották legfőképpen szem előtt, ezért nem volt igazán szükség olyan védelmi eszközre, mely korlátokat húzott a hálózatok közé, senki sem számított támadásokra. Az a feltételezés élt, hogy csak jóindulatú felhasználók csatlakoznak a rendszerhez, azonban ez az Internet publikussá tételével gyökeresen megváltozott. A Morris féreg megjelenése, valamint különböző betörési kísérletek hamar elhozták ennek a békés világnak a végét. Mivel nem volt mindenki megbízható, az összekapcsolt hálózatokban szükség volt egy védelmi eszközre. Ez az eszköz a tűzfal, mely a hálózati szegmensek határán az átmenő forgalomra előre beállított biztonsági szabályokat kényszerít.

A tűzfalak történelme 1988-ra nyúlik vissza, amikor Jeff Mogul a Digital Equipment Corporation dolgozója megjelentette az első csomagszűrő tűzfalakról szóló írást: Simple and Flexible Datagram Access Controls for Unix-based Gateways. Ebben az írásban egy szabályvezérelt csomagszűrő daemon kernelbe való integrációját írja le a szerző (magának a folyamatnak a pontos mechanizmusát is publikálta).

Dave Presotto és Howard Trickey 1989 és 1990 között a második generációs tűzfalak, az ún. circuit-level átjárók témakörében végeztek kutatásokat, végül ők hozták létre az első működőképes alkalmazásrétegbeli tűzfal modelljét. Sajnos azonban sem publikációt nem jelentettek meg, sem termék nem készült kutatásaikból.

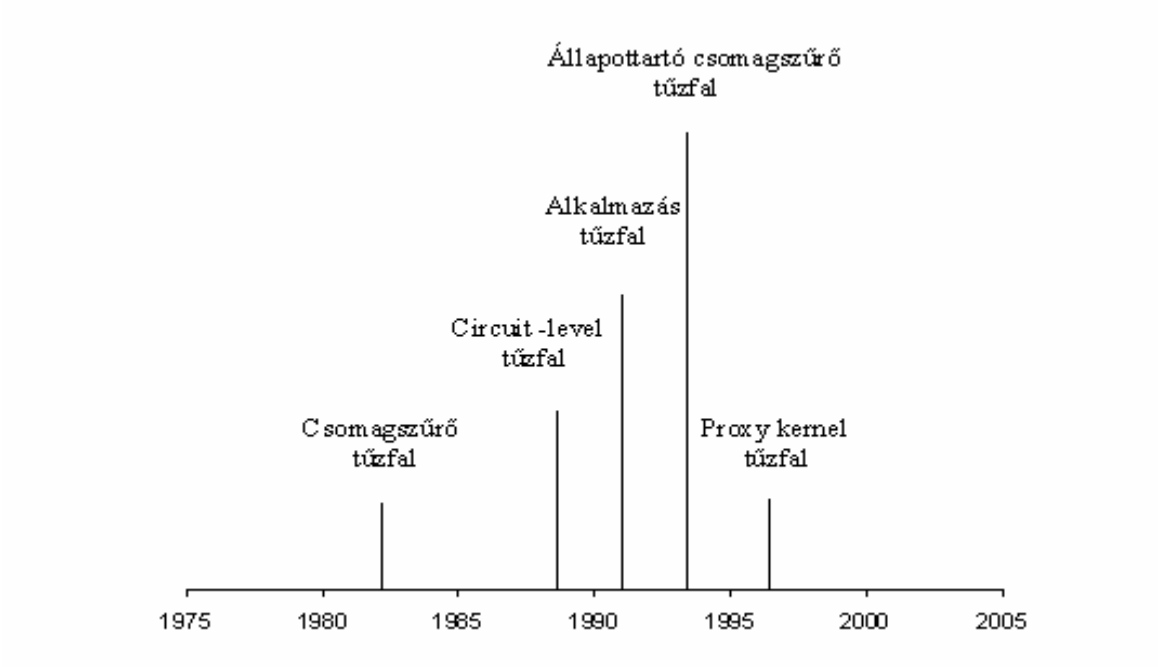
Ezt követően 1989-ben David Koblas a MIPS Computer vállalattól megalkotta a SOCKS-ot, létrehozva a „circuit-level” szintű átjárók alapjait. 1992-ben pedig Michelle D. Koblassal együtt elkészítette az implementációját is.

1991-ben többen is megjelentettek írásokat az alkalmazás szintű tűzfalak modelljéről (Gene Spafford - Purdue University, Bill Cheswick - AT&T Bell Laboratories és Marcus Ranum - Digital Equipment Corporation). Közülük Marcus Ranum munkássága kapta a legnagyobb visszhangot, amely végül az első kereskedelmi forgalomba hozott tűzfal, a DEC- SEAL kifejlesztését eredményezte, mely egy csomagszűrést és alkalmazás proxyt használó vállalati tűzfal volt.

Még 1991-ben Bill Cheswick és Steve Bellovin elkezdtek kutatásokat végezni a dinamikus csomagszűrés témakörében, azonban termék sosem lett belőle. Ettől független fejlesztésbe kezdett 1992-ben Bob Braden és Annette DeSchon az USC's Information Sciences Institute-nál. Az általuk létrehozott tűzfal a „Visas" nevet kapta, s a dinamikus csomagszűrési technológián alapult működése.

1994-ben a Check Point kiadta a dinamikus csomagszűrésen alapuló tűzfalát, a Firewall-1-et.

1996-ban Scott Wiegel a Global Internet Software Group Inc. kutatója lerakta az alapjait az ötödik generációs tűzfal architektúrájának, a kernel proxynak. Az ezen a technológián alapuló első kereskedelmi termék 1997-ben jelent meg, mely a Cisco Centri Firewall volt.

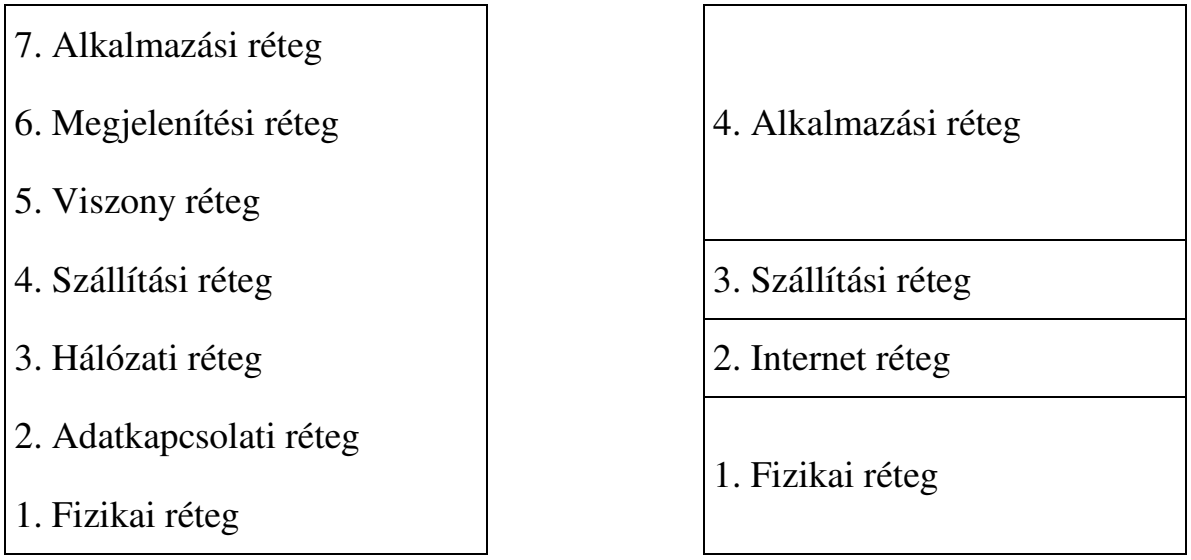


1. ábra: Tűzfalak evolúciója

[14]

2.2. Tűzfalak felépítése

A tűzfalak napjainkig rengeteg változáson mentek keresztül. Alkalmazásuk sokban függ a konkrét környezeti tényezőktől. Rendelkezésünkre állnak szoftveres és hardveres megvalósítású tűzfalak egyaránt. Ahhoz, hogy megérthessük, hogyan is működik a tűzfal, fontos átlátnunk a hálózati működés rétegek szerinti felépítését. A következő ábrán az OSI és a TCP/IP modell felépítését láthatjuk.



2. ábra: OSI és TCP/IP modellek

Az OSI modell segítségével könnyebben érhetjük meg, hogyan is működnek a különböző típusú tűzfalak. A fizikai réteg az aktuális fizikai kommunikációt jelenti, mely a különböző hardveren és médián (például Ethernet) történik. Az adatkapcsolati réteg a hálózati adatátvitel legalacsonyabb szintű részéért felelős. Ez az első olyan réteg, melyben azonosítani és címezni lehet az egyes gépeket, hosztokat. Ezeket a címeket MAC (Media Access Control) címeknek hívjuk. Egy Ethernet kártyához tartozó cím jó példa erre. A hálózati rétegben történik a WAN-okon történő adatátvitel. A rétegbeli címek az ún. IP (Internet protokoll) címek. A címek általában egyediek, de NAT esetén lehetséges, hogy több fizikai rendszer is ugyanazon a címen érhető el. A szállítási réteg speciális hálózati alkalmazásokat és kommunikációs munkameneteket (session) azonosít. Egy rendszernek bármilyen számú kiépített sessionje lehet más rendszerekkel. Ebben a rétegben a portszámokkal különíthetőek el a sessionok, melyek kiindulási és végpontként szolgálhatnak számukra. Az OSI modell további rétegei a végfelhasználói alkalmazásokat reprezentálják. Összesen 4 réteg jöhet számba egy tűzfal

működése szempontjából, ezek: adatkapcsolati, hálózati, szállítási, alkalmazási réteg. Minél fejlettebb egy tűzfal, annál több réteget tart megfigyelés alatt. Ez növelheti a konfiguráció granularitását, például lehetőséget ad felhasználó-azonosításra, hitelesítésre, speciális alkalmazások mélyebb nyomonkövetésére, natív szolgáltatások nyújtására. Ilyen beépített szolgáltatás lehet NAT, VPN, DHCP, vagy valamilyen explicit beépített tartalomszűrés. [6]

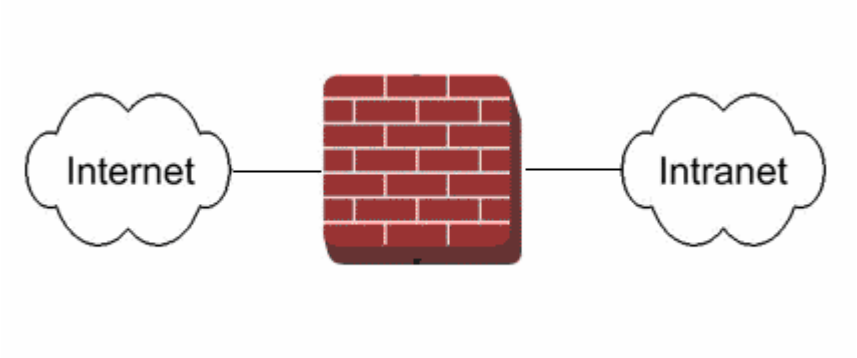
2.3. Tűzfal topológiák

A tűzfal hálózati infrastruktúrában való elhelyezkedése is meghatározza, hogy milyen képességekre van szüksége, milyen szinten kell szűrnie. Például nem mindegy, hogy perimétertűzfalként, vagy belső demilitarizált zóna elkülönítésére szolgál. Lássuk a három legjellemzőbb topológiai felosztást:

2.3.1. Bastion hoszt

Ebben az esetben a tűzfal az Internet és a védett hálózat között helyezkedik el, minden ki és bemenő forgalom rajta megy át. Ez a szerkezet a relatíve egyszerű hálózatok esetén használható (például, amelyek nem rendelkeznek semmilyen publikus internet szolgáltatással). A kulcstényező, hogy a hoszt egy egyszerű határvonalat képez, s amennyiben valakinek sikerül ezt átlépnie, a teljes belső hálózathoz hozzáfér. Ez elfogadható lehet, ha csak egy olyan vállalati hálózatot védünk, melyet csak interneten való szörfölésre használnak, viszont nem elégséges egy web vagy emailszerver esetén. Ekkor összesen 2 interfészről beszélhetünk, mely két biztonsági zónát határoz meg. Az a kérdés, hol helyezzük el az internet szolgáltatásokért felelős szervereket.

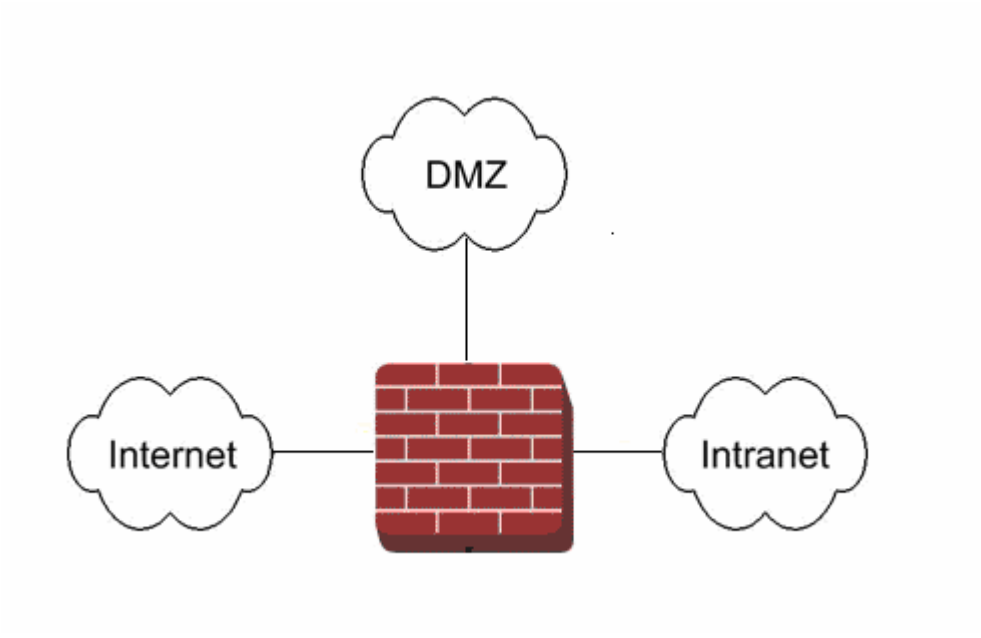
A publikus zóna esetén nem kapnak semmilyen védelmet, a privát zóna esetén pedig megnő annak az esélye, hogy a zónában lévő érzékeny rendszereket feltörjék. Amennyiben valaki a belső hálózatot akarja elérni, valamilyen módon be kell lépnie a hosztra és azonosítania, hitelesítenie magát. Mivel a bastion hoszt publikus IP címmel rendelkezik, ezért ki van téve az Internetről jövő támadásoknak, s nagyon fontos, hogy megfelelő védelemmel, frissítésekkel rendelkezzen.



3. ábra: Bastion hoszt

2.3.2. Screened subnet

A második lehetőség egy három (vagy több) lábú tűzfal használata (több mint három hálózati interfész). Ekkor a publikus hálózati szerverek egy külön hálózati szegmensbe, a demilitarizált zónában helyezkednek el. Ennek a hálózatrésznek az a feladata, hogy a külső és belső hálózatból történő kapcsolódásokat elkülönítse egymástól. A belső és külső hálózatból történő kapcsolódás a dmz-be megengedett, azonban a dmz-ből csak kifelé lehet kapcsolatot létesíteni. Ekkor mind az internettől, mind a megbízható hálózati szegmensektől (nem nyilvános fájlszerverek, munkaállomások, stb.) el vannak választva. Ha valamelyik támadónak sikerül áthatolnia a tűzfalon, a belső hálózathoz nem fér hozzá (egy jól konfigurált tűzfalat feltételezve).

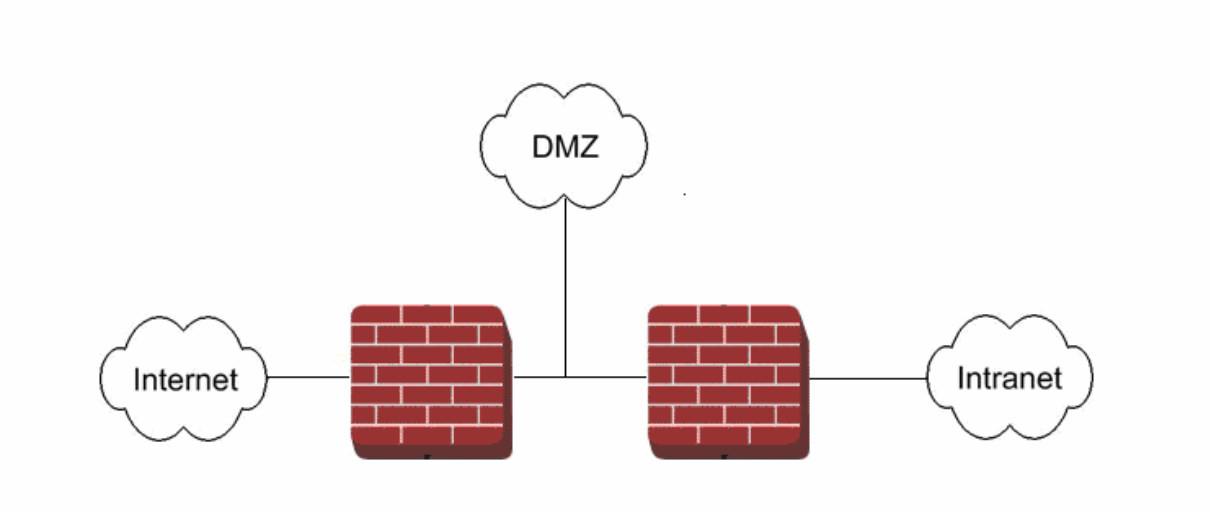


4. ábra: Screened subnet

2.3.3. Dual firewalls

A legbiztonságosabb és legdrágább megoldás ez. Ennek során két tűzfalat is használunk egy alhálózati szegmens létrehozására, s közötte helyezzük el a dmz-t. Ez egy réteges felépítést biztosít, melynek nagy előnye, hogy különböző technológiájú tűzfalakat használva kisebb az esélye annak, hogy ugyanazon biztonsági hibától, sérülékenységtől szenvednének.

A nagy teljesítményű tűzfalak lehetővé teszik hogy ne három, hanem több fizikai és virtuális interfészt különítsünk el, s ezáltal jóval több biztonsági zónát vagyunk képesek létrehozni (például lehetőség van egy szegmenst a könyvelésnek, egyet az értékesítésnek létrehozni). [7]

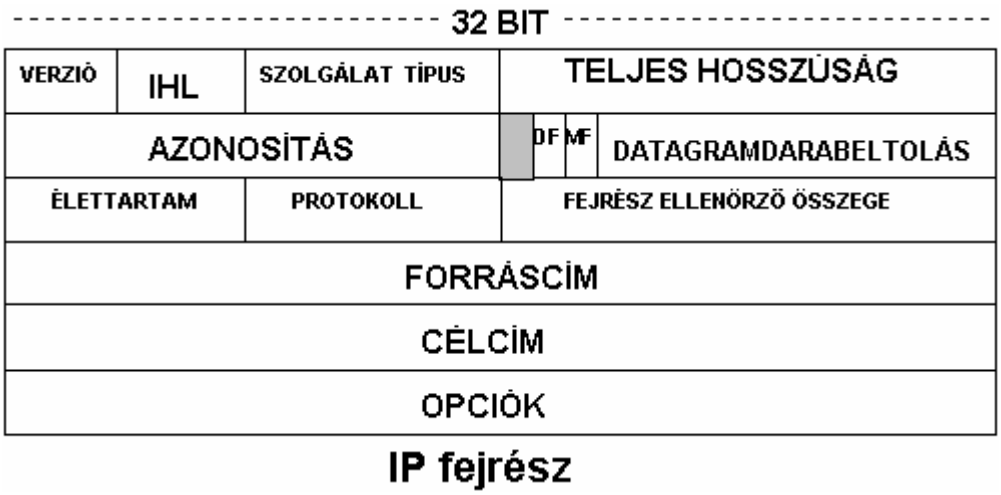


5. ábra: Dual firewall

2.4. Tűzfal technológiák

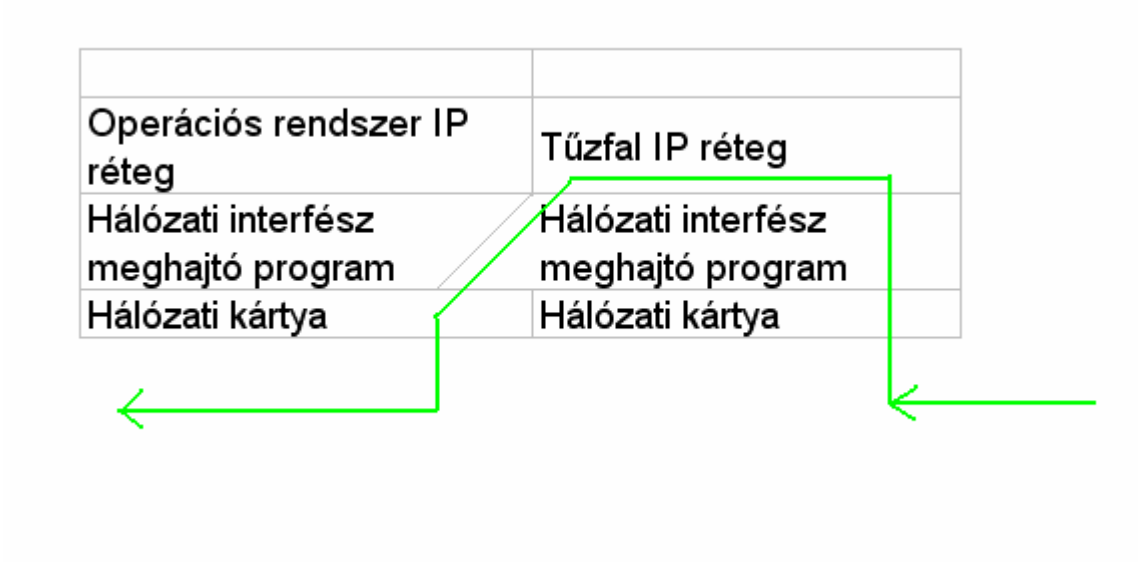
2.4.1. Csomagszűrés

Ezek a tűzfalak a TCP/IP és az OSI modell adatkapcsolati, hálózati és szállítási rétegében működnek. Általában valamilyen routerben vannak megvalósítva, hiszen ezen eszközök pont a hálózatok közötti útvonalválasztást és csomagtovábbítást végzik. Az ehhez szükséges döntést valamilyen szabályrendszer segítségével hozzák meg. Elsősorban a fejlécben található információk szerint döntenek, a csomagtartalmat nem vizsgálják, így elsősorban forrás, célcím, portszám, illetve különböző IP-flagek (például csomagtördelési adatok) játszanak szerepet a döntésben. Így elutasíthatják, továbbíthatják, naplózhatják a csomagokat.



6. ábra: IP csomag fejléce

Mostanra a legtöbb szűrésnél számításba vesznek szállításrétegbeli protokollokat is, például UDP, TCP, ICMP, IGMP. Nem veszik figyelembe, hogy az egyes csomagok milyen kapcsolatban állnak egymással. Ennek eredményeképpen gyorsak, és kis erőforrás igényűek, viszont nem tudnak olyan mély elemzést végezni. Az igazán biztonságos tűzfalak elkülönítik az operációs rendszerek és saját IP stackjüket, így védve az operációs rendszert a hálózat felől érkező támadásoktól (például: Ping of Death).



7. ábra: Tűzfal és az operációs rendszer IP stackja

Bizonyos megvalósítások lehetőséget adnak címfordításra (NAT), ekkor valamilyen segédmodul segítségével valósítják meg a fordításhoz szükséges állapottartást. A NAT (Network Address Translation), illetve PAT (Port Address Translation) két célt szolgál: egyrészt lehetőséget ad arra, hogy egy publikus IP címen keresztül több hosztot is elérhessünk

(az IPv4-es címek viszonylag kis száma, valamint belső címek elrejtésére). Ezt az IP csomag fejléc információinak (forrás-cél cím, portszám) módosításával éri el a tűzfal. Ekkor karban kell tartania táblát, mely a különböző hosztokhoz tartozó port és IP cím adatokat tartalmazza. Amikor egy hoszt valamilyen külső géphez kapcsolódik, a hálózat határán csomagjai forráscímei az átjáró IP címét kapják, s a portszám mutatja meg, hogy melyik hoszt volt valójában a küldő (amennyiben a kérésre válasz érkezik). A tűzfal elkülöníti a TCP és az UDP portszámokat is az ütközések elkerülésére. [13]

A csomagszűrök nem alkalmasak bonyolult szűrések megvalósítására, kapcsolatok nyomonkövetésére, mivel a szabályrendszer nagyon komplexszé válhat, amely a teljesítménycsökkenést is maga után vonhatja. További problémát okozhat a többcsatornás protokollok szűrése (P2P, FTP, H.323, azonnali üzenetküldés, online játékok, stb.), hiszen ekkor mégis kénytelen beletekinteni a csomag adatrészébe a tűzfal, hiszen ki kell nyernie a második csatorna portszámát. Vegyük például a DNS szolgáltatást, mely fordítást végez IP címek és domain nevek között. Amennyiben a visszaadott IP cím olyan, mely NAT-on belül található, a tűzfal kénytelen a DNS üzenetet dekódolni és megvizsgálni. Ez azt mutatja, hogy amennyiben a tűzfal a címfordítást hibátlanul kívánja végezni, kénytelen alkalmazástudatosnak leni. Ez átvezet az alkalmazásszintű szűrés felé, azonban ekkor nem történik tényleges elemzés.

Összegezve a csomagszűrő tűzfalak előnyei:

- Általában gyorsabbak, mint a többi tűzfal technológia, mivel kevesebb kiértékelést végeznek, valamint megvalósíthatóak hardveresen is.
- Egyetlen szabály is segíthet megvédeni egy hálózati szegmenst, letiltva kapcsolatokat a belső hálózati számítógépek, s konkrét külső Internet címek között.
- Nem szükséges, hogy a kliens számítógépek bármilyen módon előre konfigurálva legyenek, a csomagszűrők transzparens módon oldják meg feladatukat.
- NAT segítségével lehetőség van a belső IP címek elrejtésére a külső szemlélők, Internet felhasználók elől.

Hátrányok:

- A csomagszűrők nem tudják értelmezni az alkalmazás rétegbeli protokollokat. Ennek következtében nem tudják tiltani a különböző szolgáltatásokhoz történő csatlakozást, ha azok publikusak (például PUT vagy GET parancsok az FTP-ben), ezért kevésbé biztonságosak a többi tűzfal technológiánál.
- Állapotmentesek, nem tartanak információt sessionokről, vagy egyéb alkalmazás specifikus állapotokról, folyamatokról.
- Viszonylag kicsi változásokat tudnak az egyes csomagokon végrehajtani (IP cím átírása NAT-olás esetén).
- Nem képesek értéknövelt szolgáltatásokat végezni, mint például HTTP objektum cachelés, URL szűrés, autentikáció, mert nem képesek ezeket a protokollokat elkülöníteni.
- Nem tudják megtiltani, hogy milyen információkat kaphatnak a tűzfalon futó szolgáltatások, csak a befutó csomagok átengedéséről, eldobásáról tudnak dönteni.
- A sokféle nem triviális hálózati szolgáltatás támogatása érdekében nehéz tesztelni az elfogad/elutasít szabályokat.

2.4.2. Circuit-level tűzfalak

A Circuit-level tűzfal az OSI modell session, illetve a TCP/IP modell TCP rétegében működik. A TCP handshaking folyamatát követi nyomon, hogy megállapítsa egy-egy munkamenetről, legális-e. Egy távoli géptől a circuit-level átjárón keresztül érkező csomagról úgy tűnik, hogy az átjárótól származik, s így elrejthetőek a védett hálózat információ (IP címek, topológia). Hátrányuk ezzel szemben, hogy nem szűrik az egyéni csomagokat. Védelmi funkcióik kimerülnek a hitelesítésből adódó szűrési lehetőségekkel, valamint az erőforrások elérésének szabályozásával. Például, ha egy felhasználó ki akart jutni a védett hálózatból az Internetre, be kellett jelentkezni erre a gépre. Probléma lehet, amikor felhasználók sokasága próbál ugyanazon az erőforráshoz csatlakozni. Klasszikus értelemben nem végeznek igazi szűrést, s viszonylag bonyolult is az adminisztrálásuk, azonban mégis alternatívát jelentenek a csomagszűréssel szemben, hiszen mikor közvetlenül kapcsolódik egymáshoz a kliens és a szerver, lényegében módosítatlanul továbbítják a csomagokat, itt viszont a kommunikáció két lépcsős. Az egyik kapcsolat a hoszt és a tűzfal, a másik a tűzfal és a szerver között épül ki. Így a csomagmanipuláción alapuló közvetlen támadások lehetetlenné váltak. A közvetlen kapcsolat megszűnése után lehetőség van arra, hogy a két

kapcsolat között akár az IP csomag adatrészének szűrését végezzük. Ezek a megközelítések mutatták meg a Socks és a proxy tűzfalak számára a fejlődési irányt.

Már nem a csomag jelenti az atomi elemet, hanem a kapcsolatra koncentrálva tervezzük és valósítjuk meg hálózati határvédelmünket. Legnagyobb hátrányuk a nehéz kezelhetőség, ennek javítására jelentek meg a SOCKS és az alkalmazás tűzfalak.

Előnyei:

- Gyorsabbak, mint az alkalmazás rétegbeli tűzfalak, mert kevesebb kiértékelést futtatnak, viszont valamivel lassabbak a csomagszűrő tűzfalaknál.
- Képesek internet címek kitiltásával a belső hálózatot védelmezni.

Hátrányok:

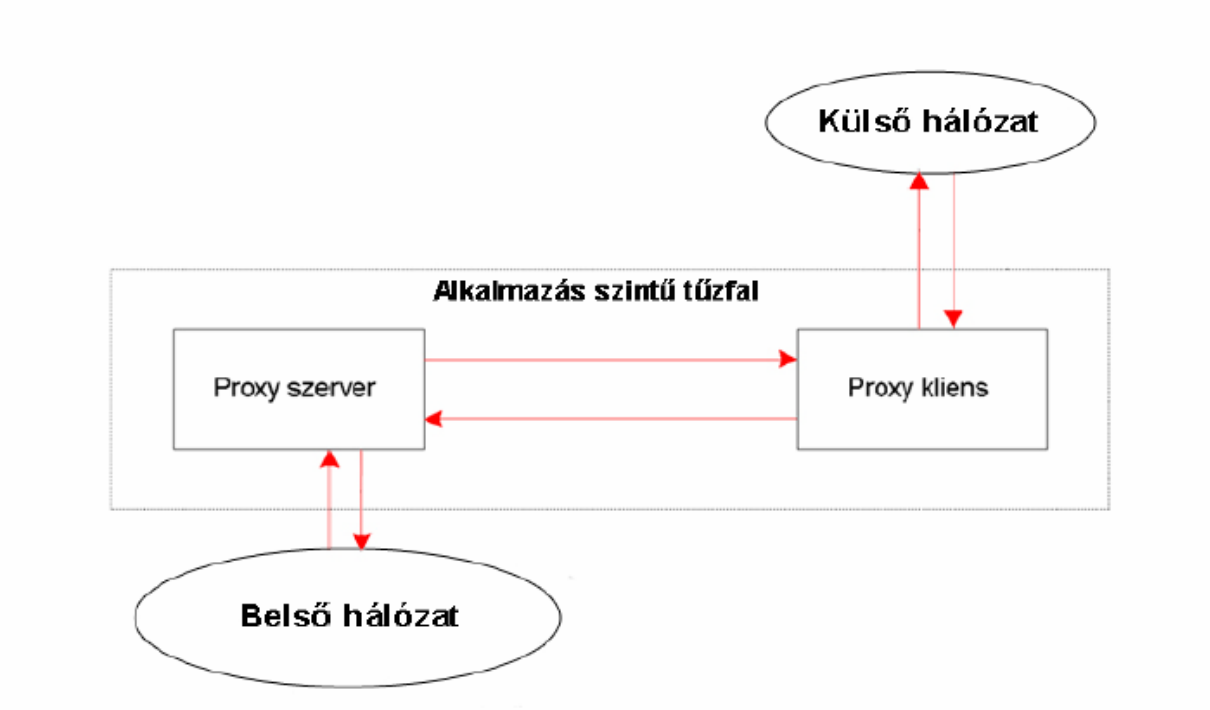
- Csak a TCP, UDP ellenőrzésére képesek.
- Nem képesek magasabb protokollokon biztonsági ellenőrzéseket futtatni.
- Limitált audit esemény generálási képességekkel rendelkeznek, de képesek hálózati csomagokat alkalmazás szintű protokollokhoz kapcsolni, korlátozott session állapotkövetés segítségével.
- Nem képes értéknövelt szolgáltatásokat végezni, mint például HTTP objektum cachelés, URL szűrés, autentikáció, mert nem képesek a protokollokat elkülöníteni.

2.4.3. Socks tűzfalak

Átmenetet képeznek a csomagszűrő és az alkalmazás proxy tűzfalak között. A Socks egy külön modulként valósul meg, mely a hálózati kapcsolatokat kezeli. Amennyiben egy program kapcsolódni kíván egy szerverhez, akkor először egy Socks proxyhoz kapcsolódik, majd a proxy kapcsolódik az elérni kívánt címhez (szerverhez). A kapcsolat létrejötte után az adatforgalom a Socks proxyn keresztül történik. Ez a működés teljesen a circuit-level tűzfalakéhoz hasonló. A működés hálózati szempontból nem, viszont az adott program szempontjából transzparens. Bizonyos programok csak natív támogatást biztosítanak a Socks használatára. Ebben az esetben ez nem tekinthető transzparensnek. A Socks legnagyobb előnye az operációs rendszerbe beépülő modul használhatósága volt, mely a szűrést hálózati szinten hajtotta végre.

A SOCKS a standard hálózati hívásokat az adott applikációban lecseréli a speciális SOCKS hívásokra. Ezután ezek a hívások egy SOCKS proxyval létesítenek kapcsolatot, egy ismert porton (1080). Ha a kapcsolatkérés sikeres, akkor a kliens egyeztet a hitelesítési módról, hitelesít, majd ha ez sikeres, küld egy relay üzenetet, melyet a szerver kiértékel, és vagy továbbítja az üzenetet, vagy megtagadja azt. Miután a kliens kapcsolatot létesített a SOCKS szerverrel elküldi neki annak a gépnek a nevét és port számát, amelyikkel kapcsolatot akar létesíteni. Ezután a SOCKS kapcsolatot létesít a távoli hoszttal, és transzparensen mozgatja az adatokat oda-vissza a két gép között. A nehézség az, hogy valakinek le kell cserélni a hálózati hívásokat a SOCKS verzióra. Jelenleg a Socks tűzfalak már nem használatosak, funkcióikat átvették az alkalmazás proxyk. [1]

2.4.4. Alkalmazás tűzfalak



8. ábra: Proxy kapcsolódás

Az alkalmazás tűzfalak, más néven proxyk hasonlóak a circuit level tűzfalakhoz, csak éppen alkalmazás és protokoll specifikusak. Nem közvetlen kapcsolat épül ki a kliens és a szerver között, a köztük folyó kommunikációt a tűzfal menedzseli. Mivel mind a ketten a proxyval kommunikálnak, a csomagszintű támadásokat minden különösebb konfiguráció nélkül képes ez az architektúra is kiküszöbölni. Az OSI réteg alkalmazás rétegében működnek, s az adott alkalmazás be és kimenő forgalmát irányítják. Semmilyen proxy nem enged át csomagokat,

ha nincs erre külön konfigurálva (például egy web-proxy nem engedi át az FTP, IRC, SNMP, vagy egyéb forgalmat). Mivel az alkalmazás réteg szintjén szűrik a csomagokat, ezért alkalmazás specifikus parancsokat is ki tudnak belőle szűrni (például HTTP:Post vagy HTTP:Get alapján), valamint a felhasználói tevékenységek és bejelentkezések is naplózhatóak vele. Viszonylag magas szintű biztonságot adnak, cserébe viszont erőforrás igényesek, s erőteljesen csökkentik a hálózati teljesítményt. Nagy hátrányuk, hogy nem transzparensek így manuális konfigurációt igénylenek a kliensgépeken.

Sokszor azonosítják az alkalmazás proxykat különböző gyorstárazó szerverekkel. Bár működésükben sok hasonlóságot mutatnak, céljuk mégis más. Mind a kettő esetében kliensek transzparens kiszolgálása a cél. Például a webcache képes lehet URL alapú, vagy a HTTP Get parancsának paramétereire való szűrést végezni. Azonban a proxyk mindig protokollértelmezéssel és parancskereséssel kezdik a szűrést. Mivel minden protokollhoz külön proxyra van szükség, fontos, hogy rendelkezzünk az összes protokollhoz szükséges proxyval, amelyet hálózatunkban használni, engedélyezni kívánunk. A szűréshez pedig ismernünk kell a protokollok speciális parancsait, paramétereit. Szerencsére nem okoz gondot a többportos kommunikáció sem, mivel a tűzfal minden alkalmazás rétegbeli információval rendelkezik. A proxy megvalósításától függően lehetővé válik a másodlagos adatcsatorna szűrése. A címfordítás kezelése sem okoz gondot, s nem szükségeltetik kiegészítő alkalmazás használata, hiszen a proxy révén az 5-7 rétegbeli protokoll session információk tárolásra kerülnek. Hátrányuk, hogy az adott alkalmazásnak támogatnia kell a proxys működést, ami nem minden protokollnál megoldott. Adott esetben kénytelenek vagyunk az egész forgalmat átengedni a tűzfalon, az esetleg csomagszűrés elvégzése után. Így biztonság szempontjából csak annyit tesz hozzá, hogy a csomagszintű támadásokat ellehetetleníti.

2.4.4.1. Transzparens proxy

A cél egy olyan működés megvalósítás volt, amely lehetővé tette, hogy a proxy funkciókat nem támogató protokollokat is kezelni lehessen. Ennek módja a tűzfalon csomagszűrés elhelyezése, melynek során a tűzfal a beérkező csomagokat elkapja, és magára irányítja. Az így átirányított kapcsolatokat proxy program fogadja. A klienseken pedig semmilyen beállítást nem kell végezni, a kliensek a célcímre küldik csomagjaikat. A csomagszűrés és a proxy együttes használata teszi lehetővé a viszonylag egyszerű kezelést és hatékony működést. A hálózati erőforrások elérése könnyebbé válik, s maga a tűzfal konfigurációja,

adminisztrációja is egyszerűbbé válik. Napjainkban egyre több összetett protokoll jelentkezik (HTTPS, IMAPS, stb.), ezáltal új technológia bevezetése válik szükségessé.

A legjelentősebb különbség a transzparens és a nem transzparens proxy között, hogy az utóbbi esetben a kliensek tisztában vannak azzal, hogy nem a szerverrel kommunikálnak. Erre jó példa a webböngészőben beállítható proxy. Sok esetben viszont szoftverkomponensek telepítésére is szükség van, ami nem mindig optimális megoldás. Transzparens esetben a csomagok protokoll fejléc-információi is igazolják a transzparens/nem-transzparens módot.

Előnyei:

- A proxy szolgáltatások megértik, s kikényszerítik a megfelelőséget a magas szintű protokollok esetén, mint például HTTP, vagy FTP.
- Az átmenő forgalomról adatokat tárol, mely kommunikáció-vezérelt, alkalmazás-vezérelt állapot, vagy akár részleges session információ is lehet.
- Lehetőség van letiltani bizonyos hálózati szolgáltatásokat, valamint hozzáférést biztosítani másokhoz.
- Az átmenő csomagok feldolgozására és megváltoztatására is képes.
- A belső hálózati és külső szerver közötti kommunikáció indirekt módon történik, így a belső számítógépek nem ismertek a külső szerverek előtt, vagyis a védett belső hálózati IP címek elrejtésre kerülnek.
- Transzparens működés révén a proxyk olyan hatást keltenek, mintha a felhasználó gépe közvetlenül a külső szerverrel kommunikálna.
- A proxy szolgáltatások képesek a belső szolgáltatásokat, valamint a belső-külső kéréseket átirányítani (például a HTTP kéréseket a HTTP szerverre, egy másik hoszton).
- Képes értéknövelt szolgáltatásokat nyújtani, mint például HTTP objektum cache, URL szűrés, felhasználói azonosítás, hitelesítés, stb.
- Jó auditálási és naplózási képességekkel rendelkezik, mivel sokféle információt képes megkülönböztetni és eltárolni az átmenő forgalomból.

Hátrányok:

- A proxy szolgáltatás a natív hálózati stack sajátjával történő helyettesítését igényli a tűzfal szerveren.

- Mivel a proxyk ugyanazon a portokon figyelnek, mint a hálózati szerverek, ezért nem lehet ugyanazon a gépen futtatni őket.
- A proxy szolgáltatások futtatása teljesítményigényes, s késleltetéseket hozhat magával. A bejövő forgalom két feldolgozáson is átmegy, az alkalmazás, s a proxy is elemzi (például az internetről befelé jövő e-mail alkalmazás kommunikál a tűzfal e-mail proxyjával, illetve a proxy kommunikál a belső klienssel).
- Általában minden olyan protokollhoz külön proxy kell, melyet át kívánunk engedni a tűzfalon, ezért a rendelkezésre álló hálózati szolgáltatások és skálázhatóságuk korlátozott.
- Az alkalmazás szintű tűzfalak nem biztosítanak proxykat alacsonyszintű szolgáltatásoknak, protokolloknak, mint például UDP, RPC.
- A proxy szolgáltatások gyakran igénylik a kliensek, vagy a kliensen lévő eljárások módosítását, bonyolultabbá téve a konfigurálást.
- Nagyon érzékenyek az operációs rendszer és az alkalmazás szintű programhibákra. A legtöbb csomagszűrő nem támaszkodik különösebben az operációs rendszer támogató mechanizmusaira, csak az eszközmeghajtó programokra. Az alkalmazás szintű tűzfalak sok támogatást igényelnek, például a NDIS, TCP/IP, Winsock, Win32m szabványos C könyvtár, stb. Ha bármelyikben hiba jelentkezik onnantól kezdve a tűzfal biztonsága kérdéses.
- Az alkalmazás szintű tűzfalak a hálózati csomag információkat figyelmen kívül hagyhatják. Így amennyiben a hálózati stack nem jól működik (amelyet nehéz validálni), akkor az olyan biztonsági ellenőrzések, melyeket a tűzfal az operációs rendszer könyvtárai segítségével végez el, hibás eredményt adhatnak. (Például egy gyakran használt utasítás a getpeeraddress().)
- A proxy gyakran jelszavas, vagy másfajta hitelesítési eljárást, igényelnek, ezáltal további késleltetést idézve elő. Másrészt a felhasználókat is idegesítheti a rendszer használatához szükséges sok-sok hitelesítési procedúra.

2.4.5. Állapottartó csomagszűrés

Ezek a tűzfalak a hagyományos csomagszűrő tűzfalak hiányosságainak kiküszöbölésére jöttek létre. A legnagyobb különbség abban leledzik, hogy immár nemcsak egyedi csomagokat használnak fel, hanem más ezzel kapcsolatban lévő csomagokat is. Ez főleg a kapcsolatorientált protokollok esetében lehet fontos. Például TCP esetén lehetőség van a

kapcsolat kiépülését megfigyelni (3-way handshaking, adatforgalom, szétkapcsolás). Ezáltal bizonyos csomagok megjelenése csak adott helyen történhet a kommunikáció során. Például TCP SYN előtt nem lehet adatcsomag. Mivel kapcsolatokat figyelünk meg, ezért nem csak az egyes IP flagek szolgálhatnak a szűrési szabályok alapjául (seq-num, ack-num, window size, stb.). Bár sokkal jobb szűrésre ad lehetőséget az állapottartó tűzfal, továbbra sem old meg bizonyos problémákat, s továbbra sem veszi figyelembe a csomag mintegy 95%-át kitevő adatrészt. Ez csak alkalmazás-tűzfallal lenne kivitelezhető, vagy egyéb beépülő modulokkal. Elmondhatjuk, hogy a csomagszűrő és az alkalmazás tűzfalak tulajdonságait kombinálják. Csomagszűrést a hálózati, tartalomszűrést az alkalmazási szinten, munkamenet validálást a session rétegben végeznek. Lehetőséget adnak a direkt kapcsolatra a kliens és a hoszt gép között, megszüntetve az alkalmazás tűzfal nem transzparens voltából fakadó problémát. Magas biztonságot adnak, s viszonylag jó teljesítményt is, azonban bonyolult szabályrendszerek segítségével teszik ezt, így a komplexitásuk konfigurálási nehézségekbe torkolhat.

2.4.6. Moduláris proxy tűzfalak

Ezen tűzfalak tekinthetőek a legújabb generációs tűzfalaknak. Transzparens megvalósításuk biztosítja a könnyű kezelhetőséget, adminisztrációt. Képesek az átmenő adatfolyam alkalmazás specifikus szűrésére, csomagszűrésre (megfelelő kiegészítő modullal). Legjelentősebb különbség a moduláris tűzfalak és a transzparens proxyk között, hogy míg a proxyk esetén minden protokollelemzést, vizsgálatot külön komponens végez, melyek nem tudnak egymással együttműködni (ezáltal sokkal inkább erőforrás igényesek), valamint sok funkciót mindegyik komponens megvalósít (kapcsolódások kiépítése, stb.), addig a moduláris proxyk képesek együttműködni sőt, kiegészítései egymásnak, így a felesleges redundanciák ki vannak küszöbölve. Egy speciális központi modul feladata lett a kapcsolatok kezelése, ha szükség van egy újabbra, akkor azt ő hozza létre, s továbbadja az adott adatfolyamot a megfelelő proxy modulnak.

A különböző protokollokhoz tartozó proxyk egymásnak adják át a feldolgozott adatokat, s a kapcsolat kezelését. Hogyan is történik mindez? Vegyük például a HTTPS-t vagy pedig a POP3 SSL-lel titkosított kapcsolatát. A kapcsolat kiépítéséért felelős proxy kiépíti a kapcsolatot, majd átadja az SSL proxynak, mely dekódolja az átmenő forgalmat, majd továbbítja a HTTP proxynak, mely számára ez olyan, mintha kódolatlanul érkezne egy

klienstől. A HTTP proxy lekezeli a kérést, esetleg elvégzi a konfigurációjának megfelelő szűrést, elemzést, majd továbbítja az SSL proxynak a végeredményt, mely titkosítja, s továbbítja. Így lehetővé vált a HTTPS forgalom HTTP szintű szűrése.

A moduláris tűzfal bizonyos moduljai fel vannak arra készítve arra, hogy átmenő forgalmuk egy részét képesek legyenek más moduloknak továbbadni, mint ahogy ők is képesek más moduloktól érkező adatok kezelésére, így a proxy modulok egymást ágyazzák be a forgalom elemzésébe. Elképzelhetőek olyan kombinációk is, amikor egy ilyen beágyazásnak nincs értelme, de a lehetőség adott. Látható, hogy ezen architektúra révén a kombinált protokollok is elemzésre kerülhetnek, mi több, a titkosítást használó protokollok is szűrhetővé válnak, s nincsen olyan fekete folt, mint az SSL titkosítást használó végpont-végpont kapcsolat. Természetesen ehhez szükséges, hogy a protokoll elemzését végző proxy ismerje az adott protokoll összes parancsát, metódusait. Ekkor lehet ugyanis képes az illegális forgalom tiltására, megváltoztatására, tartalomszűrésre. Látható, hogy a beágyazás segítségével az egyre több kombinált protokoll széles palettájának elemzésére, szűrésére van lehetőség, valamint lehetőség van nem protokoll specifikus modul alkalmazásának beiktatására (például vírusszűrésre).

2.5. Tartalomszűrés

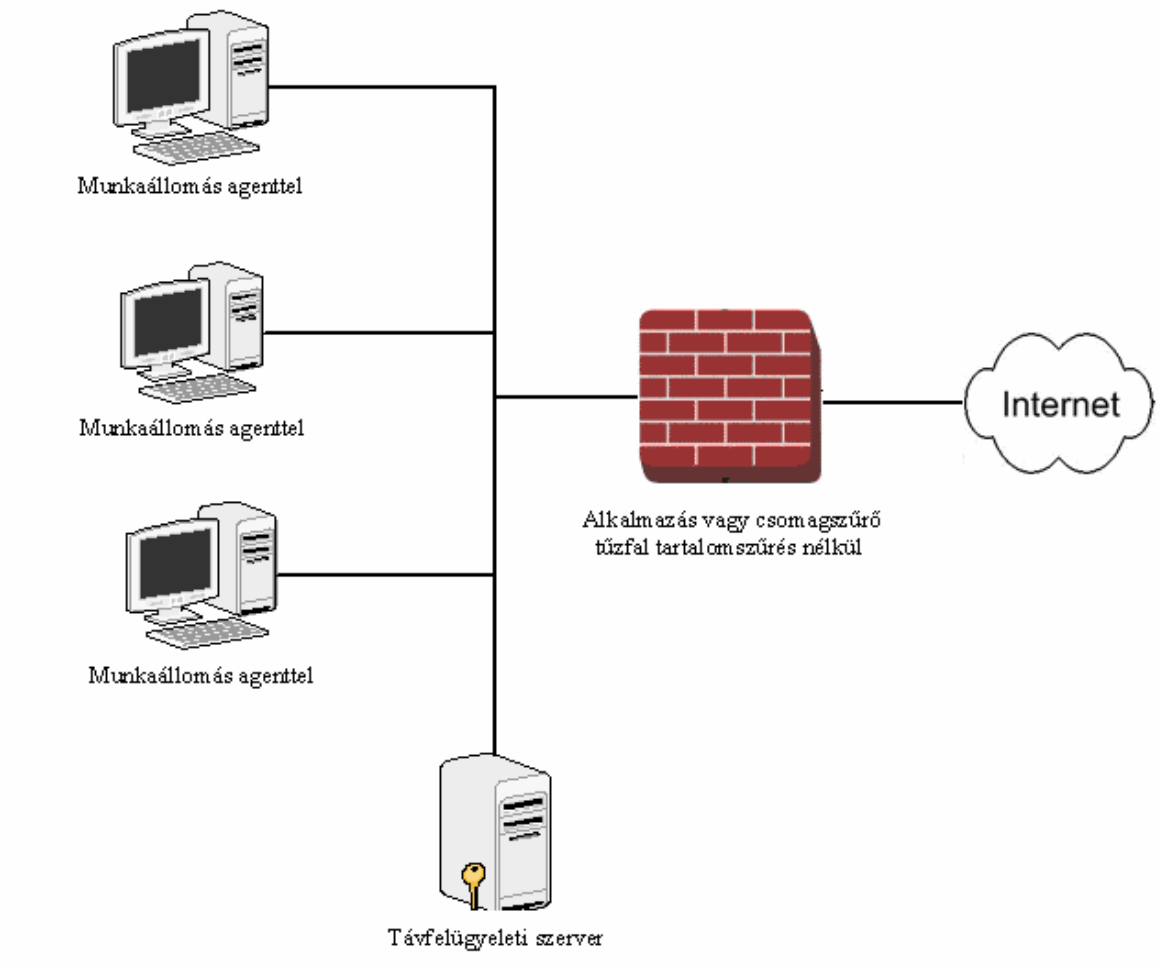
Tartalomszűrésen általános esetben az alkalmazás rétegben történő feldolgozási folyamatot értünk. Ez lehet anomália ellenőrzés, URL szűrés, vírus, spam, spyware, adware szűrés, IPS-IDS, stb. funkció.

Miért van szükségünk alkalmazás szintű szűrésre? Bár a különböző vállalatok, szervezetek nagy része használ tűzfalakat hálózataik védelmére, ezek a fajta tűzfalak nem alkalmasak bizonyos támadások kivédésére. Egy átlagos tűzfal arra van tervezve, hogy kiszűrje a nyilvánvalóan gyanús forgalmat (például SSH, SMTP csatlakozást), de átengedjen bizonyos forgalmat, tipikusan webforgalmat, a 80-as (HTTP) vagy a 443 portokon (HTTPS). Így a belső hálózati szerverek elérhetőekké válnak kívülről. A probléma ezzel, hogy sok olyan csomag is bejuttatható a hálózatba, mely valamilyen sérülékenység kiaknázására tartalmaz kódot. Ezenkívül viszont az is elmondható, hogy sok támadás a szervezeten belülről érkezik.

A webalkalmazások drámai változást hoztak abban, hogy az üzleti alkalmazások milyen módon lettek telepítve. Régebben minden alkalmazás saját belső szerveren futott, míg mostanában az üzletmenet szempontjából kritikus alkalmazások webalapúak, publikusan elérhetőek kívülről (üzleti partnerek és vásárlók számára). Így sokkal komplexebbé vált a biztonsági helyzet, a tradicionális megközelítések (VPN, PKI, IPS) bizonytalanná váltak. Ezek az eszközök a külső forgalom befelé történő mozgását tiltották ki, míg a webalkalmazások lényege, hogy a külső felhasználókat korlátozott jogokkal engedjék be a hálózatba. Az ártatlannak kinéző HTTP forgalom éppen ezért rengeteg káros dolgot tartalmazhat. Lehetségessé válik a webalkalmazás sérülékenységének kihasználása a tűzfal megkerülésének szükségessége nélkül, nem csomagszintű támadási módszereket kihasználva (csomagtördelés, túlméretezett csomagok, stb.). Ezek a támadások inkább az amúgy érvényes parancsok, sütik módosítását jelenthetik. Ráadásul, míg az OSI alsóbb rétegei jól definiáltak, addig az alkalmazás réteg sok különböző hálózati szolgáltatást nyújt. Például, míg a TCP/IP specifikációk azonosak, nincs két olyan alkalmazás, mely hasonlóan implementálná ugyanazt az üzleti logikát és technológiát. Tehát nincs egy egységes, minta-alapú megközelítés mely védene az alkalmazás rétegbeli sérülékenységek ellen.

A tartalomszűrés alapvetően két szinten valósítható meg a hálózati infrastruktúrában. Vagy alkalmazás szintű tűzfal segítségével, melyet a hálózat határára helyezhetünk el, vagy pedig a munkaállomásokon elhelyezett táv és viselkedés felügyeleti szoftverek (SurfControl, EagleEyeOS, Cisco Security Agent, stb.) segítségével. Az utóbbi esetben sokkal teljesebb körű menedzsmentet és nyomonkövetést alkalmazhatunk, mint tűzfal révén. Ekkor ugyanis lehetőségünk van a hálózati kommunikáción alapuló tartalomszűrésen kívül más szabályok betartását is ellenőrizni. Ilyen lehet például a különböző hordozható tárolóegységek, perifériák (Pendrive, Firmware, egyéb USB-s eszközök) használatát engedélyezni, tiltani, tartalmukat titkosítani. Különböző hozzáférési jogosultságokat is meghatározhatunk, az eredetileg az operációs rendszerben szereplőkön kívül (a Windows regisztrációs adatbázisához való hozzáférést, kulcsok létrehozását, módosítását, hálózat működés jellegét, dokumentumok kezelését, stb.). Ellenőrizhetjük a cég biztonsági politikájának betartását, például a telepíthető programok listáját is megadva, de történetesen egy integritásellenőrzésen alapuló programvédelem is létrehozható. Ekkor a telepített programok adott időpontbeli hash érték mentéseit vetik össze az aktuális mintával. Ekkor minden változás, melyet valamilyen külső entitás okozott (vírus, hacker, stb.), könnyedén kimutatható. Alapvetően a hoszt alapú

védelem a későbbiekben kifejtett tűzfalszintű tartalomszűrési megoldások összes funkcióját megvalósítja, azonban mivel működése valamilyen „agent” működésén alapszik, képes a teljes felhasználói tevékenység megfigyelésére (emailek, webböngészés, billentyűzetfigyelés, stb.). Ez viszont a magánszférába való beavatkozásnak is tekinthető, s súlyos „privacy” problémákat vethet fel, amit a felhasználók idegenkedve fogadhatnak. Technológiai oldaláról nézve a dolgot pedig a rendszer menedzselése, karbantartása sokkal nagyobb teher lehet, mind az IT személyzetre, mind az infrastruktúrára (hiszen ezek az agentek percenként is rákapcsolódhatnak a szerverükre), mint a tűzfalas megoldás. Egy nagyobb hálózatban érdemes lehet egy központi menedzsment szervert beállítani, amely kezeli az összes agent beállításait, változásait, s a különböző felhasználói csoportokat. Azonban az otthoni felhasználók számára is léteznek csak kliens alapú szoftverek (Netnanny).



9. ábra: Hoszt alapú tartalomszűrési rendszer

A tűzfalnak megvan az az előnye, hogy egyetlen pontban hatva érvényesíti az összes alkalmazási szintű szűrést, és kényszeríti szabályait az általa felügyelt hálózati szegmensre,

valamint csak a hálózati forgalmat ellenőrzi. Természetesen ez az ún. vállalati tűzfalakra igaz, s nem a munkaállomásokat védő személyi tűzfalakra.

Alapvetően kétféle tartalomszűrésre képes vállalati tűzfallal találkozhatunk, ha jobban körbenézünk a vállalati szegmensben. Az egyik típus az ún. „appliance" tűzfal, mely black box"-ként van megvalósítva, hiszen többnyire nem ismeretes a futtatott operációs rendszer. Erre nagyon jó példák a Cisco (Pix) vagy Symantec (Gateway Security) tűzfalai. Ennek a fajta megközelítésnek vannak előnyei és hátrányai is. Mindenképpen ide sorolható a „security by cloaking", azaz az operációs rendszer pontos tulajdonságainak hiányában nehezebb lehet kihasználható hibákat találni. Másrészt viszont a nagyobb teljesítmény érdekében történő hardverbővítés, valamint, mivel merevlemezmentes egységekről van szó, s az egész operációs rendszer valamilyen flash memóriában tárolt, ezért az egyes programfrissítések is nehézkesek lehetnek. Előnye azonban a könnyű használhatóság (könnyen hálózatba csatlakoztatható, menedzsmentje webes felületről történik leginkább), valamint, hogy a memóriaalapú működés gyorsabb, s ezért kisebb teljesítményű és energiafogyasztású hardver működhet benne.

A tűzfalak másik csoportja valamilyen teljes funkcionalitású számítógépre telepített, operációs rendszeren futó szoftver. Erre a célra felállítva működhet egy szolgáltatás, vagy a teljes operációs rendszer is. Ez utóbbira jó példa az állapottartó csomagszűrésre beállított, Linuxot futtató tűzfal hoszt. Sok egyéb operációs rendszer-tűzfal létezik, hogy csak egy párat említsek a nagyobbak közül: ISA 2005 (Internet Security and Acceleration) Server, vagy a CheckPoint Firewall-1. Ezen megoldások kétségtelen előnye a könnyebb skálázhatóság, valamint a hardver újrafelhasználhatósága. Hátránya viszont a sokkal átláthatatlanabb konfiguráció.

A hagyományos csomagszűrőkkel ellentétben az alkalmazás tűzfalak minden esetben feldolgozzák a csomagok adatrészét valamilyen szinten. Például az FTP protokoll aktív módban való működéséhez a vezérlőkapcsolat kiépülése után a tényleges adatforgalom a PORT paranccsal megadott számon történik. Az alkalmazásintelligencia nélkül a kliens tűzfal nem tudná, melyik portot nyissa meg. Így vagy az aktív FTP átvitelt kellene tiltani, vagy az összes portot FTP adatátvitelre (ami hatalmas nagy biztonsági rés lehet). Hasonló események történnek a több portos protokollok esetén, valamint olyan általánosan használtak estén, mint például RTSP, DNS, VoIP.

A tartalomszűrő tűzfalak (moduláris, transzparens és nem transzparens proxy) elméletileg képesek a teljes átmenő adatforgalom alkalmazás szintű szűrésére. Ehhez, mint már előbb említésre került, ismernie kell az adott protokoll összes szabványos utasítását és metódusát, valamint képesnek kell lennie az átvitt adat elemzésére. Az előbbi a mélyprotokoll elemzés, az utóbbi az ún. content vectoring megvalósítása.

A mélyprotokoll elemzés azért nehézkes, mivel sok hálózati eszköz nem ellenőrzi a protokollok betartását, a szabványoknak megfelelő megvalósítást. Így a protokollt sértő, de az adott alkalmazás valamilyen sérülékenységét kihasználó csomagokat lehet létrehozni és eljuttatni a támadott gépre. Ennek következtében az adott protokoll összes szabályát ismerő proxy képes minden illegális kommunikációs kísérletet megszakítani. További előnye, hogy jóval részletesebb információt kaphat egyes eseményekről, s ezáltal sokkal kifinomultabban is reagálhat.

A tartalomszűrés során valamilyen nem kívánatos erőforráshoz, anyaghoz való hozzáférés korlátozását végezzük. Gyakran előfordul, hogy nagy cégeknél a kritikus, érzékeny információkhoz nem mindenki juthat hozzá. Ugyanígy igaz, hogy nem lehet tetszőleges dokumentumot elküldeni a céges hálózaton kívülre. Ezen kívül számtalan más dolog található az interneten, melyhez célszerű korlátozni a hozzáférést. Ilyenek például a pornográfia, chat, illegális multimédia anyagok (filmek, zene), szerencsejáték, stb., melyek akár a cég ellen törvényi eljárások kezdeményezését is maga után vonhatja. Emellett nagyon sok olyan kártevő van, mely az adott oldalt meglátogatva, a böngésző rossz beállításait használva települ, s fertőzi meg a gépeket. Másfelől a dolgozók figyelmét is eltereli a munkáról a sok, nem kimondottan munkakörükhöz kapcsolódó oldal olvasása. Különböző fájlkiterjesztések is kitilthatóak a hálózatból. Ezáltal a jellemzően káros végrehajtható fájlokat, vagy más potenciálisan veszélyes fájltípusokat tarthatunk távol a rendszertől.

A HTTP és HTTPS forgalom szűrése napjainkban már valós igény. A titkosított adatfolyamnak megvan az a veszélye, hogy a tűzfalak, IDS-ek minden beavatkozás nélkül átengedik, s a végpont-végpont átvitel során a célgépen okoz problémát, fertőzést, kártékony kódok feltelepítését. Ezenkívül nagyon alkalmas érzékeny információk kiengedésére, kiszivárogtatására.

A tartalomszűrés csökkentheti a veszélyeket a nem megfelelő böngészés esetén, azonban fontos tudni, hogy ez a fajta védelem nem helyettesíti az adminisztratív szabályok által biztosított védelmet.

A content filtering segítségével lehetőség van csak ellenőrzött eszközök használatára szorítani a dolgozókat (például webmail, azonnali üzenetküldők forgalmának tiltásával).

Az ActiveX vezérlők régóta a legkockázatosabb dolgok közé tartoznak. Hiába vannak esetleg aláírva, vagy más valódinak látszó hitelesítéssel ellátva, ezek könnyen megtéveszthetik a felhasználókat. Viszont ezeket a vezérlőket szinte bármilyen nyelven implementálhatjuk, s így szinte korlátlan funkciók megvalósíthatóak velük. Ráadásul a számítógép összes erőforrásához korlátlanul hozzáférnek. Például hozzáférhetnek a helyi fájlrendszerhez, kapcsolatokat létesíthetnek, fájlokat vihetnek át egyik hosztról a másikra. [5]

A következő listán lehetséges szűrési feltételek szerepelnek:

- Web forgalom: IP cím, vagy tartomány alapján való szűrés a 80-as porton, kulcsszavak, kategóriák (erőszak, rasszizmus) alapján. Lehetséges különböző heurisztikus, mesterséges intelligencia módszereket használni, így a html kód és az oldal tartalmának átvizsgálása után automatikus kategóriába sorolás történhet, s létrehozható fekete, illetve fehér lista a látogatható oldalakra. A szűrés során megkülönböztethetjük a felhasználói csoportokat is, mikor kiengedjük őket az internetre, vagy pedig időbeliséget is meghatározhatunk (például munkaidőben nem látogatható weboldalak listája). Nem utolsó sorban alkalmas lehet a puffer túlcsordulási támadások szűrésére is.
- Valósidejű URL elemzés: URL és metatagjainak átkutatása bizonyos szavak után, esetleges átirányítás a tiltott oldalakról a megfelelő figyelmeztető oldalra, vagy módosítás.
- Java és ActiveX vezérlők vizsgálata.
- Magic-byte-ok keresése a fájltípus megállapítására, adott letöltés visszautasítására.
- Ismert vírusminták utáni keresés.
- Grafikus dimenziók meghatározása a kéretlen reklámok blokkolása érdekében (ads, pop-up),
- SSL dekódolás, forgalomelemzés

- Email tartalom vizsgálata, spamek, phising kiszűrése.
- Javascript vizsgálat.
- Kifelé kommunikáló programok adatforgalmának elemzése (dokumentum védelem).
- Alkalmazás specifikus adatforgalom vizsgálat, hozzáférés protokollok (IPsec, Radius, 802.1x),
- Betörési minták
- QoS, hang és video streaming vizsgálata, DRM ellenőrzések
- Azonnali üzenetküldők figyelése
- Spyware, adware szűrése
- Anomim proxyk kitiltása
- Fájl megosztás, fájlcserélés kitiltása

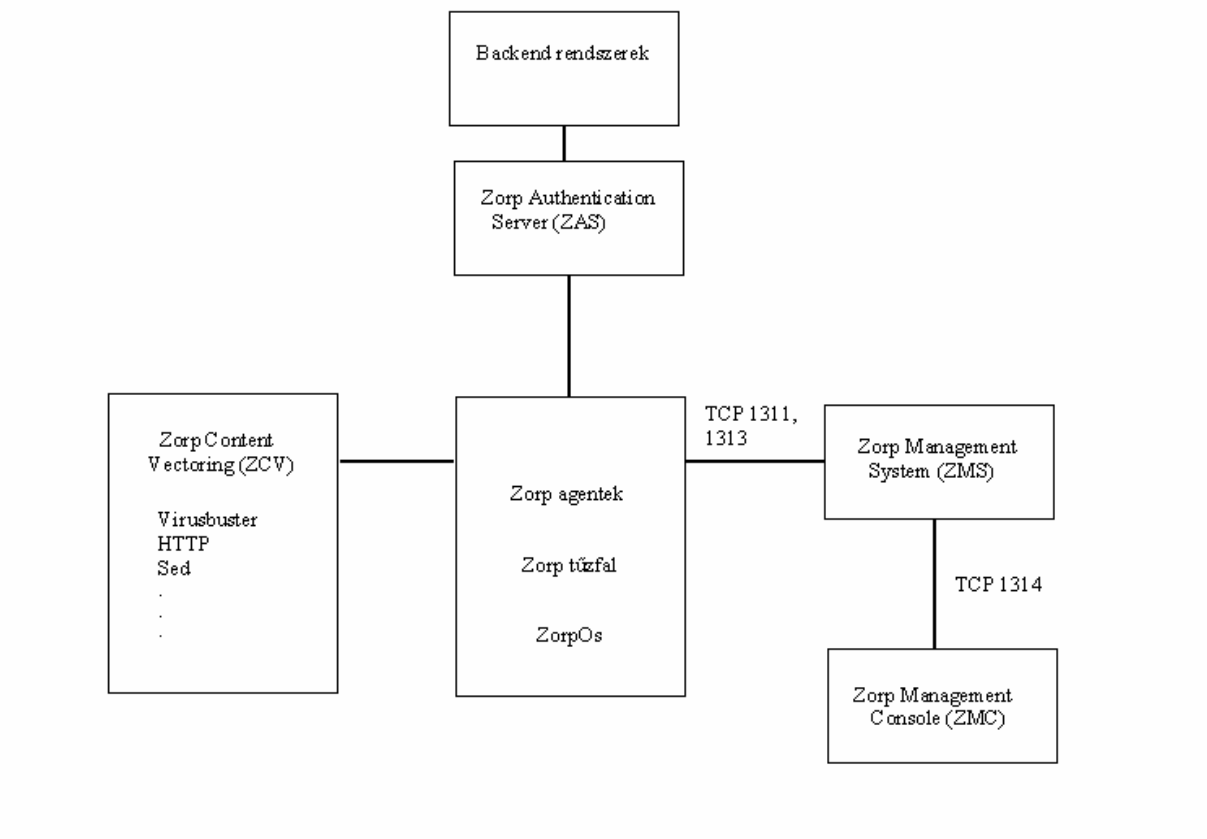
Fontos tudni, hogy minél több ilyen szkennelést végzünk, annál erőforrás-igényesebb lesz a szűrés. További gondot jelenhet a sok hamis pozitív találat, egy idő után szinte használhatatlanná teszi a hálózat használatát.

Emellett még nagyon sokféle kifinomult logikát építhetünk bele a tartalomszűrésbe, például a megjelenített színekből megpróbálhatjuk összerakni, hogy mennyi bőrszín jelenik meg a képen, s bizonyos százalék felett a tiltott pornográf anyagnak minősíthetjük a dolgot, még ha a többi szűrésen át is jutott. [15]

3. Zorp technológia

A Zorp egy alkalmazás szintű tűzfal, mely nem csupán egy telepíthető szolgáltatásként működik, hanem egy előrekonfigurált, megerősített Debian alapú operációs rendszeren fut (ZorpOS). Ezáltal nem csak alkalmazás szintű szűrést tud végezni, hanem ki tudja használni az operációs rendszer sajátosságait, s képessé válik bizonyos fokú csomagszűrés végzésére is (IpTables, Netfilter). Így olyan hibrid tűzfallá válik mely jóval biztonságosabb, mint egy egyszerű alkalmazás tűzfal. Például a bejövő forgalmat azonnal csomagszinten ellenőrzi, majd csak az érvényes portokra beérkező forgalmat továbbítja a magasabb szintű alkalmazás proxyknak. Mivel speciális célú, a tűzfal funckiókat támogató komponensei vannak, ezért munkaállomásként, vagy szerverként nem annyira használható, bár teljesen funkcionális operációs rendszeren fut, s így természetesen vannak beépített natív szolgáltatásai (Bind, NTP, Postfix, DNS, stb.). Valójában feltelepíthetőek különböző szerver csomagok a tűzfalra, azonban ez nyilvánvalóan gyengíti a biztonságot.

A csökkentett operációs rendszer előnyei, hogy csak a szükséges szoftver komponensek települnek, ezáltal meglehetősen biztonságos platformot biztosít a tűzfal szoftver számára, csökkenti a lehetséges szoftver hibákból adódó támadási felületet, valamint megerősített linux kernelt tartalmaz, mely a létező összes publikált foltot magába foglalja. A Zorp tűzfal a következő szoftver komponensekből áll:



10. ábra: Zorp komponensek

3.1. Zorp tűzfal

A tűzfal komponens végzi a szabályrendszereiben, szűrési szabályokban megadott forgalomszűrést. Ezt mind a négy, a tűzfalak által elérhető rétegben végzi. A beérkező csomagok először a Zorp állapottartó csomagszűrőjén mennek keresztül, mely szabályainak megfelelően látja el őket route információkkal, esetlegesen módosít paramétereiken (TOS, TTL, IP forrás cím (transzparens működés miatt), stb.), majd megfelelőség esetén az adott proxyhoz továbbítja a csomagot.

A proxyk alapvetően mélyprotokoll elemzési képességgel rendelkeznek, s választható, hogy transzparens vagy nem transzparens állapotban fusson, az igényeknek megfelelően (Tproxy kernel folt támogatása révén). A szoftver magas rendelkezésre-állást biztosító High Availability (HA) fürt, illetve titkosított virtuális magánhálózat (VPN) kialakítására alkalmas modulokkal is rendelkezik. Továbbá képes az általa nem ismert protokollokat átengedni a tűzfalon, az ún. plug proxy segítségével, mely segítségével bármely egy porton kommunikáló kliens-szerver kapcsolatok felügyelhetőek. (MYSQL, VNC, Microsoft Terminal Service, SMB/CIFS, SYSLOG, IPP, RSYNC, stb.). Ekkor természetesen nem lehet szó protokoll

validálásról. Ezen kívül képes az alprotokollok elemzésére, az egyes protokollok feldolgozás után továbbíthatják a beágyazott részeket a további proxyk, vagy a feldolgozó modulok számára. Ezeket a modulokat a ZCV valósítja meg, arról, hogy pontosan milyen modulok is tartoznak ide, a későbbiekben bővebben szó lesz.

A Zorp tűzfal képes több tűzfal példányt futtatni, melyek egymástól teljesen függetlenek lehetnek, s más-más forgalomért, protokollért lehetnek felelősek, így a különböző adminisztrációs követelményeknek is megfelelhetnek. Ezen túl a több tűzfalpéldány lehetőséget ad arra, hogy hiba esetén egy további példány tovább is fusson, s elássa a tűzfalfunkciókat (hiszen a tűzfal példányok egymástól függetlenül elindíthatóak, és leállíthatóak). Természetesen multiprocesszoros rendszerekben (SMP, HyperThreading) több példány használata teljesítményjavulást is okozhat (Mivel a teszt során nem ilyen rendszert használtam, ezért a több példány futtatásáról lemondtam). Azonban mivel minden tűzfal példány külön szálon fut, melyek mindegyike külön memória helyet igényel, gyorsan kifogyhatunk a memóriából is.

3.1.1. Zorp Management System (ZMS)

Központosított menedzsment interfészt kínál a tartományába tartozó tűzfalak karbantartására, konfigurációjára. Ezeket a beállításokat saját XML alapú adatbázisában tárolja. Legnagyobb ereje, hogy egyszerre lehet ugyanazon hálózati környezetbe (tipikusan egy vállalat hálózata) tartozó tűzfalakat konfigurálni, s nem kell egyenként, node-ról node-ra állítani őket. Ezen kívül adatbázisában tárol egy biztonsági mentést az általa menedzselt tűzfalak beállításairól, hiszen egy konfigurációs módosításnál a ZMS adatbázis változik, s az agentek segítségével módosulnak az adott tűzfalnak beállítások. Ehhez előbb fel kell tölteni az adott hosztra az új konfigurációs állományt, majd újra kell futtatni vagy tölteni az adott szolgáltatást. A ZMS lehet ugyanazon, vagy külön gépen is, ezt lényegében csak a rendelkezésre álló erőforrások befolyásolják. Az első esetben sincsen közvetlen kapcsolat a ZMS és a tűzfal között, a kommunikáció mindig agent alapú a kérdés csak az, hogy Unix socketet vagy IP csomagokat használ. A ZMS-ben rengeteg féle beállítási lehetőség található, lényegében a tűzfal szolgáltatáshoz kapcsolódó összes konfigurációs lehetőség. A ZMS-t közvetlenül egy kliens programmal (ZMC) érhetjük el, de lehetőség van a tűzfalat a központi menedzsment rendszer használata nélkül is konfigurálni, ebben az esetben viszont jóval körülményesebb a dolog. Beállítható, hogy a menedzsment szerver milyen IP cím tartományból fogadjon el csatlakozási kéréseket, amely szintén csökkenti a jogosulatlan hozzáférések számát.

3.1.2. Transfer and monitoring agents

A ZMS nem közvetlenül kommunikál a Zorp tűzfallal, vagy az alatta lévő operációs rendszerrel. Ezt ún. menedzsment ügynökök (agentek) segítségével történik, melyek a tűzfalat futtató hoszton települnek, s ezek után képesek titkosított (SSL) kommunikációt folytatni a ZMS-sel. A kapcsolat létesítése és a hitelesítés tanúsítványok segítségével történik. Az ügynökök a felelősek a tűzfal konfigurációs, egyéb hozzátartozó információinak gyűjtéséért, a ZMS felé való küldéséért, valamint a konfigurációs parancsok fogadásáért, végrehajtásáért. A kommunikáció TCP-n keresztül a 1311 (konfigurációs fájlok átvitele, végrehajtása (zms-transfer-agent)), és 1313 (monitorozás, nyomon követési feladatok (zms-monitoring-agent)) portokon történik, azonban ez adott esetben átállítható más portokra is. Általában a kommunikációt a ZMS kezdeményezi, de a konfigurálás során ez megfordítható, ha szükséges.

3.1.3. Zorp Management Console (ZMC)

Ez a grafikus interfész a ZMS kezeléséhez, lehetővé teszi a távoli elérést és menedzsment, természetesen titkosított SSL kapcsolaton keresztül (hiszen az interneten történő üzenetváltás során egy harmadik fél információkat szerezhetne a tűzfal beállításairól, amely nem kívánatos). Bár igazából a ZMS elérhető e komponens nélkül is. A két komponens a TCP 1314-es porton keresztül kommunikál egymással. A ZMC mindig a ZMS adatbázist változtatja meg, sohasem közvetlenül a tűzfal konfigurációs fájljait.

3.1.4. Zorp Authentication System (ZAS)

A Zorp tűzfal nem rendelkezik olyan adatbázissal, mely hitelesítési információkat (jelszó, felhasználói név, hozzáférési jogosultságok) tárolna, így olyan backend (apache htpass fájl, RADIUS szerver, LDAP szerver, Active Directory, PAM) rendszerekkel kommunikál, melyek ezeket az adatokat elérhetővé teszik. A ZAS köztesrétegként működik a tűzfal és ezen backend rendszerek között, ha hitelesítés történik, akkor a tűzfal kapcsolódik a ZAS-hoz (saját konfigurációjának megfelelően), amely viszont a megfelelő backend-hez kapcsolódik. A hitelesítési adatokat kinyerve a ZAS végzi a hitelesítés (egyszerű jelszó alapú azonosítás, Kerberos, X.509 tanúsítvány, kihívás-válasz protokollok) teljes folyamatát, s az eredményt visszaadja a tűzfalnak. A komponens elsődleges célja egypontos azonosítás és hitelesítés

nyújtása, a mögötte lévő adatbázis rendszerek elfedésével, lehetőséget adva multi-vendor környezetek integrációjára.

3.1.5. Zorp Content Vectoring (ZCV)

A ZCV nem csak egy tartalomszűrést végző motor, hanem egy olyan keretrendszer, mely segítségével harmadik fél tartalomszűrési megoldásait menedzselhetjük és konfigurálhatjuk, s integrálhatjuk a tűzfal működésébe egy egységes interfészen keresztül. A modulok a tűzfaltól függetlenül futnak, még csak ugyanazon a gépen sem kell lenniük (ekkor azonban a ZorpOs, mint operációs rendszer tovább is szükséges), a Zorp el tudja küldeni a vizsgálandó adatokat a megfelelő feldolgozó hosztnak a vizsgálathoz szükséges paraméterekkel, konfigurációs beállításokkal. Például a víruskereső modul mindig fájlszűrést végez, de más paraméterekkel végezheti az email csatolmányok vizsgálatát, mint a HTTP-n keresztül letöltött fájlokét. Ugyanakkor a protokollok is meghatározhatják a szűrés módját. Például a HTTP forgalom vírusszűrésen, tartalomszűrésen eshet át, s az összes kliens oldali szkript eltávolításra kerül. SMTP esetén vírus-szűrésen és spam-szűrésen eshetne át a tartalom.

A content vectoring hasonlónak tűnhet, mint az alkalmazás szintű protokollelemzés, melyet a Zorp az alkalmazás proxykon keresztül végez. Azonban lényegi különbség, hogy a proxyk a protokollok elemeit analizálják, nem magát a transzferált adatokat.

3.2. A content vectoring főbb céljai

Vírusszűrés: A legklasszikusabb és legismertebb formája a tartalomszűrésnek. Az átvitt fájlok vizsgálata, annak ellenőrzéseképpen, hogy nem tartalmaznak semmilyen olyan programot, mely károsíthatja a felhasználó gépét, vagy az infrastruktúrát. A legtöbb vírusvédelmi szoftver a trójaiak és adware szűrésére is képes.

Spam-szűrés: e-mailek szűrése (SMTP forgalom többnyire) a kéretlen, vírusos levél törlésére. Kliens oldali szkriptek tiltása a HTML oldalakban, hiszen virtuálisan bármilyen operációt végezhetnek a kliens gépen.

Általános HTML szűrés, kulcsszó alapján, vagy fekete/fehér lista alapján történő szelekciója a tiltott, engedélyezett anyagoknak, vagy csak egyszerűen a minden napos munkába nem illeszkedő oldalak kiszűrése.

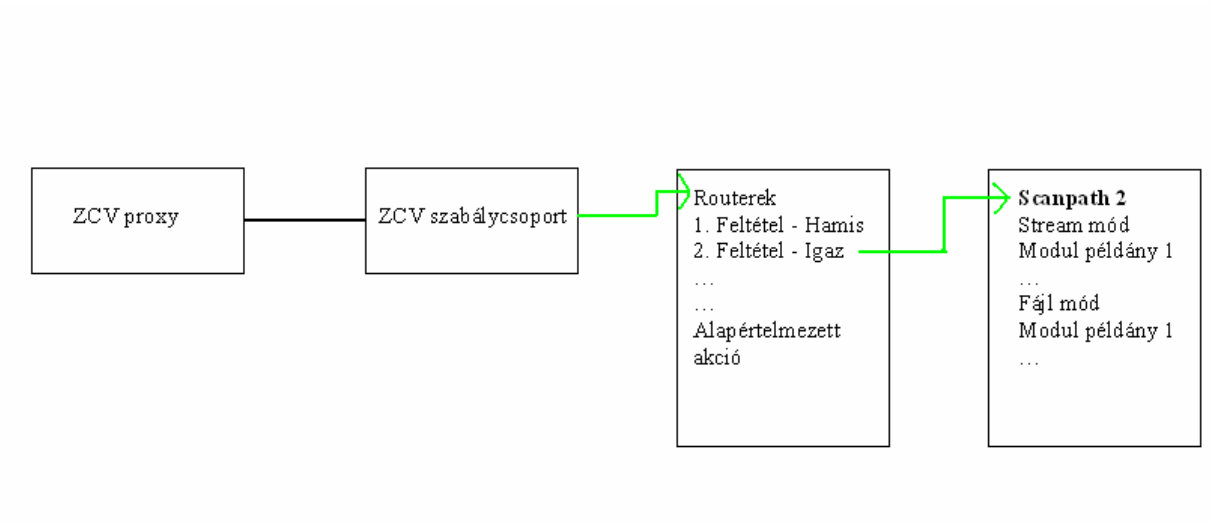
A CV két megközelítést alkalmazhat: fájl és stream-alapú szűrés. Fájl esetén komplett objektum (fájl) szükséges az ellenőrzés végrehajtásához, például vírus vagy spam szűrésnél. Streameknél folyamatos adatfolyamot figyelünk meg (például egy teljes weboldal letöltése), s a tiltott tartalmat töröljük, vagy módosítjuk (javascript, képek, stb.).

A visszautasított objektumok többnyire törlésre kerülnek, azonban vannak környezetek, ahol ez nem engedhető meg, így ezekben karanténba kerülnek, ahol nem okozhatnak bajt, s ideiglenesen tárolhatóak, amíg nem ellenőrzik, hogy nem tartalmaznak-e fontos információkat. Ennek oka, hogy a detektálási megoldások nem 100%-osak. Produkálhatnak hamis pozitív találatokat, így kritikus adatok esetén vigyázni kell, nehogy elvesszen belőlük az átvitel során. Az éles rendszerekben éppen ezért ajánlott több tartalomszűrő modult is alkalmazni, melyek megvizsgálják ugyanazon forgalmat, csak éppen más-más módszerekkel. A ZCV teljesen támogatja az ilyenfajta többes content vectoring végzését.

3.2.1. Hogyan működik pontosan a Content vectoring a Zorp tűzfalban?

Mindenfajta tartalomszűrést a moduloknak egy-egy példánya végezhet. Ezek a példányok elsősorban konfigurációs beállításaikban különböznek, hiszen mindegyik saját absztrakt moduljának a leszármazottja. Egy absztrakt modul tartalmazza a modul felépítését, működését, de az inicializációs paraméterek, attribútumok nélkül. Ezek a példányosítás során kapnak értéket. Minden modulnak éppen ezért sajátos tulajdonságai vannak, melyeket a későbbiekben részletesebben tárgyalok.

Egy Zorp proxy egy ZCV szabálycsoportnak küldhet adatokat további elemzésre. Ez a szabálycsoport határozza meg, hogy a szcenáriók során, melyek különböző feltétel-akció párokon alapszanak, milyen feldolgozási lépések szükségesek. Ez a döntés az objektumok meta-információin és a ZCV által gyűjtött adatokon alapszik. A feltétel bármilyen információ lehet, melyet a Zorp/ZCV ismer: IP cím, MIME típus, stb. Az akció vagy valamilyen alapértelmezett művelet (elfogadás, eldobás, továbbítás), vagy egy ún. „scanpath” lehet, mely a content vectoring modul példányokat (modulok és beállításaik egy adott szcenáriónak megfelelően) adja meg, melyek a forgalmat vizsgálják. A szabálycsoportoknak van alapértelmezett scanpath beállításuk, azonban a routerek a szabálycsoportban mást is választhatnak adott kondícióktól függően.



11. ábra: Zorp proxy működés

A képen egy példa, a Zorp proxy elküldi a forgalmat a megfelelő ZCV csoportnak, mely kiértékeli a router kondiciókat, majd az elemzés következik a megfelelő modul példányok által. Ezután a feldolgozott adatok visszakerülnek a Zorphoz.

A scanpath-ok a modul példányok és azok paramétereinek listáját (karantén beállítások, stb.) tartalmazzák. Ez lehet fájl vagy stream modul egyaránt, azonban fontos tudni, hogy mindig a stream modulok dolgozzák fel először az adatot.

3.2.2. Scanpath opciók

- Karantén mód: Meghatározza, mikor kell egy fertőzött fájlt a karanténban tárolni. Mindig az eredeti fájl kerül ide.
    - Always: Minden elutasított objektum tárolása.
    - When rejected: A javíthatatlan objektumok tárolása
    - When modified or rejected: A fertőzés mentesítés sikeressége esetén is az eredeti fájl kerül karanténba.
    - Never: Karantén tiltása
- Oversize threshold: Maximum fájl méret, ennél nagyobbak nem kerülnek vizsgálatra teljesítmény okokból.
- Trickle mode: A tartalomszűrés nem végezhető részlegesen meglévő fájlokon, a teljes fájlra szükség van a vizsgálathoz. A klienshez való küldés csak akkor kezdődik meg, ha vírusmentesnek bizonyult. Ebből kifolyólag a kliens egy darabig nem kap

semmilyen adatot, majd hirtelen nagyon sokat. Ez kis fájloknál nem probléma, hiszen ekkor érezhető késleltetés nélkül megtörténik az átvitel. Nagy fájlok esetében ez gondot okozhat, akár a kliens alkalmazás időtúllépést jelezhet, s bonthatja a kapcsolatot. Ugyanezt a jelenséget okozhatja, ha nagy a sávszélesség különbség a tűzfal két oldalán lévő hálózatok között. Ezek elkerülése végett az ún. trickling használható, mely során a tűzfal apró csomagokat kezd el küldeni, hogy fenntartsa a kapcsolatot.

- Disabled: A trickling le van tiltva.
- Percent: A küldött adatok mennyisége az adott objektum méretétől függ. A klienshez csak új adat érkezésekor küld a tűzfal újabb csomagokat, ezeknek az összmérete a fogadott adatok X százaléka lehet.
- Steady: Fix mennyiségű adat áramlik a klienshez, s a küldés a belső késleltetésnek megfelelően kezd el átküldödni. Amennyiben az egész fájl letöltődik és feldolgozódik az adott intervallum alatt, úgy nem történik trickling használat.

Célszerű a százalékos módszert használni, mivel ekkor a legnagyobb a valószínűsége, hogy a tricklingből adódó késletetés nem lesz érzékelhető a rendszerben. A művelet jellegéből adódik, hogy ez csak fájl módban működik.

3.2.3. Proxy modulok

Egy proxy modul egy olyan szoftverkomponens, mely bejövő adatfolyamot dolgoz fel, s utána a kimenetére ennek eredményét adja. Minden proxy modul esetén lehetőség van egy (vagy több) előredefiniált, alapbeállításokkal rendelkező proxy példány használatára. Azonban, ha speciális beállításokat kívánunk használni, például fejléc információkat módosítani a kérésekben, akkor az adott proxy absztrakt osztályából leszármaztatott osztályt kell használnunk, ahol megadhatjuk ezeket a tulajdonságokat. Ezekben az osztályokban jelennek meg a proxyk által implementált alacsonyszintű tulajdonságok.. Ezek a leszármaztatott osztályok azok, akik valójában példányosíthatóak, s meghívhatóak egy scanpath-ben. Természetesen lehetőség van saját leszármaztatott osztály létrehozására is, vagy a meglévők módosítására.

A modul példányok a ZCV-nek bizonyos content vectoring feladatokra konfigurált elemei, melyek saját paramétereikben különbözhetnek egymástól. Néhány modulban léteznek globális opciók, melyek az adott modul összes példányára érvényesek.

A proxy komponensek felelősek az átmenő adatforgalom mélyprotokoll analíziséért. Eközben az adott forgalom protokoll validálását is elvégzik a megfelelő RFC-k szerint, s ha nem megfelelő akkor módosítja, tiltja a kapcsolatot. Például a HTTP forgalom szűrése az RFC 2616 és az RFC 1945 szerint történik. Erre jó példa lehet a CODE RED féreg, mely egy speciális URL kérést használt fel a terjedéshez, mely egyébként nem volt érvényes az RFC-k szerint, de az IIS szerverek mégis kiszolgálták.

A proxy osztály az alacsonyszintű proxyt, s a hozzátartozó beállításokat tartalmazza. Az osztályok felelősek a csomagok adatrészének vizsgálatáért, a többi komponenssel (routerek, listenerek) együttműködve a kommunikációs csatornák monitorozását végzik. A legtöbb osztály protokoll specifikus, azaz tudtában vannak az adatfolyam feldolgozása közben a protokoll-információknak.

A Zorp jelenleg 24 modullal rendelkezik, ezek: Finger, FTP, HTTP, IMAP, Ldap, LP, Mime, MSRpc, NNTP, POP3, PSSL, Radius, RSH, SIP, SMTP, SQLNet, SSH, Telnet, Tftp, Whois, Virusbuster, Kaspersky, Plug.

A proxyk beállítások tetszőlegesen megváltoztathatóak, kérések parancsok, válaszok mindegyikéhez külön parancsok, utasítások rendelhetőek. Minden protokollhoz egy policy hash tartozik, mely a lehetséges eseményekhez rendel hozzá egy akciót. Ez lehet elfogadás, tiltás hibaüzenettel, vagy nélküle, kapcsolatbontás, valamilyen döntéshozó eljárás meghívása. Az utolsó reakció más szinten valósul meg (proxy vagy python), mint az előzőek, ezért némileg lassabb lehet a kiértékelés, mint az előző esetekben.

A legtöbb proxy támogatja a másodlagos sessionök létrehozását, azaz több kapcsolat használja egyszerre ugyanazon proxy modul példányt (ugyanazt a szálat). Ez jelentősen csökkentheti a terhelést, mivel nem kell minden kapcsolathoz új proxy példányokat létrehozni, s ehhez memóriatöbbletet használni. A Zorp egy új kapcsolat elfogadásakor megpróbál keresni egy megfelelő proxy példányt, amelyik képes fogadni az új sessiont, s ha nem talál, csak akkor hoz létre újat. Megadható a proxy példány osztályában, hogy milyen

kritériumokkal fogadjon el másodlagos sessionöket. Jelenleg a Plug, Radius és Sip modul képes erre.

Amennyiben egy proxy további analízist tart szükségesnek, úgy az adatokat verem használatának segítségével küldheti tovább. A vizsgálat után a verembe rakott proxy vagy program visszaadja az adatokat az eredeti proxy-nak, amely folytatja az átvitelt.

A vermek használata elsősorban beágyazott protokollok esetén, vagy víruskereséshez szükséges. Meghatározható, hogy ellenőrzéskor az összes adat elemzésre kerüljön, vagy csak bizonyos protokoll elemekhez kapcsolódóak (például HTTP GET kéréshez). MIME információk is átkerülhetnek elemzésre, azonban míg az adat mezők az elemző modulokban is módosulhatnak, addig a fejléc csak a legfelső proxyban modulban változhat meg.

Lehetőség van ún. „program verem" használatára is, ekkor valamilyen feldolgozó modul helyet az adatfolyam a standard inputra irányítódik át, s ennek során valamilyen program, szkript futtatható le rajta, majd a standard outputról az eredményt kapja meg az eredeti proxy. Ez lehetőséget ad arra, hogy valamilyen saját fejlesztésű elemző modulon futtassuk át az adatfolyamot, s nincs szükség, hogy a Zorp Content Vectoring interfészéhez illesszük az egyedi modulunkat.

A következő két modul a tesztelés szempontjából is fontos, hiszen ezeket felhasználva, s különböző konfigurációikban futtatva végeztem el a méréseket.

3.2.3.1. Plug proxy

A plug proxy szolgál arra, hogy az olyan protokollokat is kezelni lehessen, melyek jelenleg nem rendelkeznek proxykkal a Zorp-ban. A plug proxy egyszerűen továbbítja a forgalmat a tűzfalon keresztül anélkül, hogy bármilyen ellenőrzést is végezne. Képes mind TCP, mind UDP csomagokat továbbítani célcímükre. Olvassa a kliens oldali kéréseket, majd létrehoz egy kapcsolatot a szerver oldalon is. Azonban ez a protokoll is előszűrt (az első védelmi réteget adó csomagszűréssel), s a csomagszintű támadásokat kivédi, valamint más modulokba átirányítható további feldolgozásra. Történetesen itt csak a vírusvédelmi, HTML moduloknak,

illetve a sed-nek lehet értelme. Ha belegondolunk ez az a proxy, mely a circuit-level tűzfal funkciót megvalósítja.

3.2.3.2. HTTP proxy

Ez az a modul, mely a HTTP (Hypertext Transfer Protocol) protokoll elemzésére képes. Mivel ez az Internet egyik legfontosabb és leggyakrabban használt átviteli protokollja, nagyon fontos, hogy megfelelő vizsgálatnak vethessük alá. Segítségével nagyon sokféle webes tartalomhoz férhetünk hozzá. Mivel a protokollban nincsen megszorítás arra nézve, hogy milyen típusú fájlokat vihetünk át segítségével, ez az egyszerű szöveg fájltól kezdve a legösszetettebb multimédiás tartalomig változhat. A modul transzparens és nem transzparens módon is képes működni. Az utóbbi esetben lehetőség van az SSL protokoll használatának kérelmére, ami annyit jelent, hogy a kliens a proxyval HTTP-n keresztül kommunikál, míg a proxy a szerverrel HTTPS-en. Lehetőség van fejléc és adatrész információk változtatására, új sorokat lehet beilleszteni, törölni lehet őket, vagy változtatni értékeiken. URL átirányításra, vagy visszautasítására is lehetőségünk van, vagy pedig bizonyos adatmódosítások végrehajtására.

A Zorp különbséget tesz a szerver és a proxy kérések között. A szerver kérések általában a böngészők által küldött kérések, melyek közvetlenül kommunikálnak a HTTP szerverekkel. Ezek a kérések a szerver gyökeréhez képest relatív útvonalat (/pelda/index.html) tartalmazzák, és a hoszt cím, pedig megadja a kiszolgáló virtuális szervert.

Proxy kérés esetén a böngésző egy HTTP proxy-val kommunikál. Ez a kérés teljesen specifikált URL-t (http://example.com) tartalmaz. Bár teljesen nyilvánvalóan detektálható megkülönböztetés nincs a kettő között, a kéréseket máshogy kezeli transzparens (szerver kéréseket vár) és nem transzparens (proxy kéréseket vár) módban a proxy. Nem transzparens módban a proxy a direkt FTP kéréseket (ftp://rapid.eik.bme.hu) azonnal átfordítja.

Most pedig vessünk egy pillantást azokra a modulokra, melyek a content vectoringot valósítják meg a Zorpban.

3.2.4. ZCV modulok

3.2.4.1. VirusBuster (vbuster) modul

A vbuster modul a VírusBuster víruskereső modulját használja, mely csak fájl módban működik. A következő paraméterek állíthatóak:

- Disinfect: Fertőzött fájlokat megpróbálja eltávolítani.
- Scan packed: Tömörített fájlok között is keres.
- Scan suspicious: Gyanús fájlok vizsgálata (az adatbázisban nem szereplő vírusok kiszűrésére).
- Infected action: Ha nem fertőtleníthető a fájl, akkor milyen akciót hajtson végre.
- Heuristic scan level: Heurisztikus keresés szintjei:*OFF*, *NORMAL*, *HIGH*.
- Scan level: Keresés érzéknysége: *FAST*, *STRICT*, *FULL*.
- Remove all macros: Makrók eltávolítása a fájlokból.
- Remove all OLE objects: OLE objektumok eltávolítása a fájlokból.
- Archive max size: Ennél nagyobb méretű fájlokat nem vizsgál meg..
- Archive max ratio: Max tömörítési arány vizsgálata. Ha a tömörített fájl kicsi, de kitömörítve nagyon nagy, az gondot okozhat a kliens gépen.
- Continue on database load error: Amennyiben a vírus adatbázis nem elérhető, az összes fájl átengedendő (adatvesztés elkerülésére)

3.2.4.2. Kaspersky modul

Az előzőhöz hasonló fájl módban működő víruskeresési modul.

3.2.4.3. HTML modul

A HTML modul segítségével szkriptek és tag-ek szűrésére van lehetőség. Fájl és stream módban egyaránt működik. A következő opciókkal rendelkezik:

- Enable JavaScript filtering: Minden JavaScript és eseménykezelő (pl: *onclick*, *onreset*, etc.) kiszűrése.
- Enable ActiveX filtering: Minden ActiveX komponens eltávolítása, eltűnteti az *applet* tags és a *classid* prefixeket.

- Enable Java filtering: Eltávolítja az összes Java kód referenciát. (például: *java:*
- and *application/java-archive, applet* tag).
- Enable CSS filtering: Eltávolítja a CSS elemeket.
- Filter HTML tags: Tetszőleges szűrési minták adhatóak meg, amelyeket eltávolít a HTML forrásból.

3.2.4.4. Általános stream szűrő modul (sed)

A sed modul fájl és stream módban is képes működni, a paraméterként kapott streamben helyettesíti a reguláris kifejezéssel adott mintákat. Működése hasonló a Unix sed parancsához. Viszont egy példány sok szűrőt tartalmazhat. Ezek sorrendje pedig tetszőlegesen változtatható. A következő lehetőségekkel rendelkezik:

- Regular expression: Azt a stringet definiálja, mely után a keresést végzi a bitfolyamban a következő beállítási opciók léteznek:
    - Replacement: Helyettesítés, az egyező minta a megadott karaktersorozattal lesz helyettesítve. Ez például tiltott oldalak esetén lehet hasznos.
    - Global: Ekkor az összes mintát helyettesíti a megadottal. Egyébként csak a legelső találatot.
    - Case insensitive: Kisbetű-nagy betű érzékenység.

A ZCV rendelkezik néhány olyan opcióval is, mely a hardver kihasználtságot befolyásolja. Ezek közül a legfontosabb a memória és a merevlemez-használat. Ezeket az erőforrásokat főként ideiglenes objektumok, kicsomagolt tömörített fájlok, stb. tárolására használja.
A következő memória beállítások állnak rendelkezésre:

- Max. disk usage: A maximum merevlemez méret, amit a ZCV használhat.
- Max. memory usage: A maximum memória méret, melyet a ZCV használhat.
- Low and high water mark: A ZCV lehetőségei szerint mindent memóriában tárol. Amennyiben eléri a „high water mark” méretet, úgy el kezdi kiírni a merevlemezre az adatokat, egészen, amíg nem lesz „low water mark” méretű.
- Max. non-swapped object size: Az olyan objektumok, melyek ennél kisebbek, sohasem kerülnek a merevlemezre kiíráskor.
- Content-type preview: A MIME típus meghatározására kiolvasott bájtok szám az adott objektumból. Minél magasabb ez a szám, anál megbízhatóbb a detektálás.
- [8], [9]

4. Teljesítmény analízis

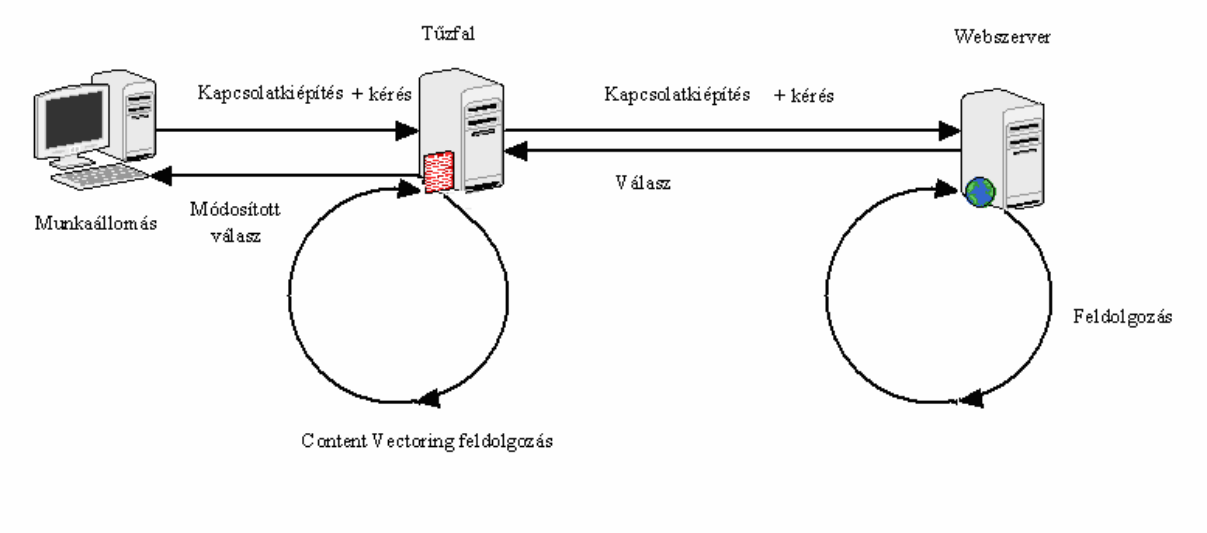
Fontos kérdés, hogyan érhetjük el, hogy megfelelő színvonalon szolgáljuk ki a felhasználókat, de a szűrés hatékonysága se szenvedjen csorbát, s az adott biztonsági szint is megmaradjon a hálózatban. Mint tudjuk a biztonság és kényelem egymásnak ellentmondó fogalmak, így kénytelenek vagyunk bizonyos engedményeket tenni mindkettő javára egy hálózati infrastruktúra kialakítása során.

Megfelelő sávszélesség illetve feldolgozási kapacitás szükséges ehhez. Ahhoz, hogy megfelelően tudjuk skálázni a hálózatot, ismernünk kell a kiszolgálás mérőszámait. Egy rendszerben rengeteg olyan adat található, mely önmagában nem, csak valamilyen korrelációban érvényes. Meg kell találnunk azokat az adatokat, melyek lehetővé teszik a rendszer finomhangolását, hogy adott hardver adottságok mellet a szoftverteljesítmény optimális legyen.

4.1. Mérőszámok

Alapvetően elmondható, hogy két szűk keresztmetszetet fedezhetünk fel a teljesítmény vonatkozásában. Egyik az erőforrások elfogyásával járó szolgáltatásminőség csökkenés. Ez akkor léphet fel, amikor például a webszerver, vagy a tűzfal nem képes már több kapcsolatot kiszolgálni, hiszen kilencven valahány százalékos memória és CPU kihasználtságon fut. Ennek következtében dobja kapcsolatokat, a beérkező csomagokat, lassan (vagy gyorsan) elérhetetlenné válik. Ez megfelel egy DoS (Denial of Service) támadás hatásainak.

A másik esetben a tűzfal és a mögötte lévő kiszolgáló rendszer működik, azonban nagyobb késleltetéssel adja vissza a kért oldalakat. Ennek oka elsősorban a beiktatott feldolgozási lépésekben keresendő. Mivel ezek általában valamilyen szekvenciális műveletsort hajtanak végre, növelik a ráfordított kiszolgálási időt. A következő ábrán egy általános tartalomszűrési megoldás esetén bejövő késletetési időket számolhatjuk.



12. ábra: Késletetés generálodása a rendszerben

A legtöbb fellelhető metodika és irodalom a késleltetést és a teljes átviteli sávszélességet tartja a tűzfalakat jellemző mérőszámoknak, erre több mérési módot, és adatmennyiséget ajánlanak:

- Egy időben létesíthető és működtethető kapcsolatok száma (concurrent-connection):
- A legtöbb alkalmazás kapcsolatorientált protokollokat használ, így az a kérdés, hogy mennyit tud kiszolgálni adott időn belül fontos. Mivel ezen protokollok elsősorban a szállítási rétegben található TCP/IP protokollra épülnek, elsősorban ennek változásait érdemes mérni. A kapcsolat kiépülése azt is feltételezi, hogy képes adatforgalmat is bonyolítani, tehát egy HTTP objektum átvitelére sor kell, hogy kerüljön a teszt során.
- 1 másodperc alatt létesített kapcsolatok száma (connections-per-second): Az a sebesség, mellyel az új kapcsolatok kezdeményezése történik. Ekkor azt mérjük le, hogy 1 másodperc alatt hány új kapcsolatot lehet felépíteni és lebontani. Azért nem elég csak egy kapcsolat kiépülési idejét venni, mert az újabbak létrehozása egyre jobban és jobban megterheli a tűzfal rendszer erőforrásait, így viszonylag jól mérhető a CPU használat hatékonysága. Ez a mérőszám csak a kapcsolat-orientált protokollra érvényes, azaz a TCP-re.
- Kapcsolatlétesítés és lezárás leggyorsabb, leglassabb, átlagos ideje: A mérőszám az előzőhöz hasonlóan alkalmas lehet az erőforrások hatékony kihasználását mérni. Az idő statisztikák igazán a terhelés függvényében információértékűek. Ezenkívül amikor egy kapcsolat teljes kiszolgálási idejét mérjük, a kapcsolatépítés-bontás idejének levonásával megkaphatjuk a tényleges feldolgozási időt.

- Kapcsolatlétesítési adatmennyiség (minimum- maximum- átlagos): Az adat mennyiség, mellyel a hosztok között kiépül egy kapcsolat. Ez az áteresztő képességbe nem számítandó bele, révén nem hasznos forgalom.
- Kapcsolatfenntartási adat mennyiség: Ahhoz, hogy egy kapcsolat ne szűnjön meg, időről-időre bizonyos vezérlő csomagok váltása szükséges. Ez szintén nem számít bele a hasznos adatforgalomba.
- Áteresztőképesség: Az a sebesség, mellyel a tűzfal kapja, illetve küldi az adatokat, bit/másodpercben, vagy csomag/másodpercben mérve, amikor még nem jön létre csomagvesztés az átvitel során. Ezt folyamatosan, állandó sebesség tartásával. Bitek esetén az IP csomag számít (fejléc és hasznos teher), más mezők, mint például adatkapcsolati rétegbeli fejléc, és keret nem. Ebbe beleérthető az elutasított, megengedett, illegális forgalom is, tesztelés során csak a megengedett forgalmat érdemes nézni. Fontos a bit továbbítási sebességet a terheltség szintjéhez, a forgalom céljához, elosztásához is mérni.
- Továbbítási sebesség: Az a sebesség, mellyel a beérkezett forgalmat továbbítja a tűzfal a célpont felé bit/másodpercben, vagy csomag/másodpercben mérve. Az interfészre beérkezési pillanat, valamint a cél interfészre való érkezés pillanata között eltelt időt számítjuk. A különböző alkalmazás szintű szűréseken átmenve ez igazán fontos mérőszáma az alkalmazás rétegbeli tűzfalaknak, hiszen itt tudhatjuk meg a feldolgozási műveletek által okozott késleltetést. Itt szintén csak IP csomaghoz tartozó bitek számítanak a mérés során.
- Protokoll késletetés: Egy protokoll-kérés és válasz között eltelt idő. Ezt különböző terhelési szinteken nézve kapjuk meg a feldolgozásból, vizsgálatokból adódó késleltetés idejét, mely egyben a különböző erőforrásoktól való függést is mutatja.
- TTFB (Time-to-first-byte): Az eltelt idő, míg a kliens megkapja a válasz első bájtját.
- TTLB (Time-to-last-byte): Az eltelt idő, míg a kliens megkapja a válasz utolsó bájtját. A TTFB és TTLB között eltelt idő szintén alkalmas a feldolgozásból adódó késletetés pontosabb detektálására, hiszen amennyiben nem egyenletes az átvitel, a csúcsok mutatják hol vannak kisebb nagyobb szűkületek az erőforrás hozzáférésben. [2], [3], [4]

Magasabb szintű mérőszámokat pedig, az alkalmazás szintű protokollok kiszolgálási idejének mérésével kaphatunk:

- FTP/HTTP átvitel teszt: az átvitt adatmennyiség, a beállított szabályok, csatlakozott felhasználók, stb. függvényében, az illegális forgalom százalékában (beállított szabályok függvényében). Ekkor különböző méretű HTTP objektumok kerülnek átvitelre, s számon tartjuk a kért, s kiszolgált objektumok számát.
- FTP/HTTP stressz teszt: maximum átvitel.
- HTML válasz idő URL/oldalanként, tranzakcióként.

[16], [17]

A tesztnek van még egy fontos része, még pedig a DOS támadás esetén való viselkedése. Ekkor meg kell határozni, hogy az egyes mérési típusok hogyan változnak egy meglehetősen magas terhelés esetén. Ezeket az eredményeket az alap mérési eredményekkel kell összehasonlítanunk. Ezek a támadások jellemzően a TCP protokollt használhatják, így a szóba jöhető támadás a TCP SYN, ekkor a támadó hoszt TCP SYN üzeneteket generál, és küldi el az áldozatnak. A megtámadott válaszol ezekre az üzenetekre és várja a végső üzenetet, hogy befejeződjön a kapcsolat kiépítése. De mivel a válasz cím nem érvényes, ezért soha nem választ kapni, és így értékes erőforrásokat köt le, amíg nem timeout-ol (tipikusan 1 percig). Megfelelő gyorsasággal generálva ezeket az üzeneteket random IP címekkel, a támadó feltöltheti a szerver kapcsolódási sorát és megakadályozhatja TCP-t használó szolgáltatások működését (www, fájltranszfer, stb.). Nincs könnyű módja a támadó visszakeresésének, mivel az IP címek generáltak. A támadás során fontos statisztikai tényező lehet a SYN csomagok gyakorisága is. Fontos kérdés az, hogy ekkor erőforrás hiányban beenged-e illegális forgalmat a tűzfal, vagy továbbra is jól zár. Bár ez a pont inkább már a sérülékenység-vizsgálathoz kapcsolódik.

4.2. Teljesítménymérés

Az előző fejezet tanulságai alapján a teljesítménymutatók mindkét fontos mérőszámának mérésére sor kerül. Tehát késletetési és adatátvitel sávszélesség adatokat fogok mérni és összehasonlítani az operációs rendszer különböző erőforrás kihasználtságának függvényében. Ez az OSI réteg több szintjén is megtörténik, azaz egy szállítás rétegbeli (TCP) és egy alkalmazás rétegbeli (HTTP) protokollt fogok mérni. A TCP/IP protokoll azért kerül kiválasztásra, mivel a leggyakrabban használt protokoll a hálózatokon, ráadásul ha az üzenetvesztés nem lehetőség, akkor mindenképpen ezt kell használnunk, az UDP kevésbé biztonságos és megbízható, mivel kapcsolatmentes protokoll. Ezen kívül a másik mért

protokoll is erre épül, így alkalmunk lehet megfigyelni, hogy lassul-e az átvitel magasabb rétegszinten alkalmazott szűrések miatt. A mérés során négy féle tesztelési típus különböztethető meg. Először is egy általános referenciamérés történik, mely a működési alapértékeket határozza meg. Ekkor minden proxy és content vectoring modul beállítás nélkül futtatjuk a terhelési teszteket. Ekkor csak a tűzfal gépen lévő iptables és routing beállításával történik a forgalomszűrés. Ez egy szimpla csomagszűrő tűzfalnak felel meg. Ezután a modulok bekapcsolásával, s a ZCV modulokról szóló fejezetben található paraméterek változtatásaival végzem a mérést. Ekkor várhatóan csökkeni fog a teljesítmény, kérdés milyen arányban. Két féle proxy modul tesztet kívánok végezni. Először a plug modulon engedek át HTTP forgalmat, majd a HTTP modulon magán ugyanezt. Ebből megtudhatjuk, hogy mekkora különbség lehet egy sima proxy kapcsolódás és átvitel, illetve a közbeiktatott mély-protokoll elemzés között. Legvégül pedig a ZCV bekapcsolásával a valós idejű tartalomszűrést tesztelem a Zorp HTML moduljának segítségével.

Idáig nem beszéltem a hardver korlátokról, melyek szintén befolyásolhatják a tűzfal teljesítményét. Azt, hogy az egyes tesztesetekben a memória, CPU, merevlemez kapacitás vagy hálózati interfész sávszélessége a szűk keresztmetszet, szintén jól nyomonkövethető. Ehhez egy erőforrás figyelő programot használok a tűzfal oldalán (vmstat).

Nem elhanyagolható tényező, hogy milyen fájlokat kérünk le az adott szerverről. Egy kisebb fájlnak a vizsgálata, könnyebb és gyorsabb lehet, míg egy nagyobbé lényegesebben több memóriát igényelhet. Ebből kifolyólag a mérés során sok féle méretű fájlt próbálok ki. Mivel a HTML modul tesztelem, ezért elsősorban text/html típusú fájlokat használok majd, de sor kerül egyes esetben tömörített fájl használatára.

4.2.1. A tűzfal operációs rendszerének korlátai

- CPU kihasználtság: Mennyire elfoglalt a processzor, és melyek azok a folyamatok, amelyek a legtöbb gépidőt elveszik? Az elfoglalt processzor jelentheti egy rossz, vagy hibásan konfigurált erőforrás jelenlétét is (pl. hálózati interfész)! A felesleges folyamatok, valamint a tűzfal működése szempontjából nem fontos programok kiszűrhetőek, a telepítés után megmaradt automatikus indítású programokkal együtt. Egy közel 100%-os CPU kihasználtsági statisztika utalhat a processzor szűk

keresztmetszetére. A különböző szálak nyomonkövetése pedig megmutathatja, hogy az egyes elemzési feladatokkal mennyi időt tölt a tűzfal, vagy a ZCV-t futtató gép(ek).

- Memória kihasználtság: Ha egy olyan operációs rendszeren fut a tűzfal, mely virtuális memóriát használ, meg kell győződjünk arról, hogy a tűzfal nem lapoz (írja ki a memória lapokat a diszkre) folyamatosan. A lapozó fájl mérete és kihasználtsága is mérvadó lehet, egy magas százalékú kihasználtság esetén célszerű lehet növelni ennek méretét. Ha a rendszer fizikai memóriát használ, meg kell, hogy mérjük a használt és szabad memória statisztikáit, és annak puffer vonatkozásait.
- Diszk kihasználtság: Győződjünk meg róla, a tűzfal nem fut ki a szabad helyből. Például kérdés, hogy van-e elég helye a tűzfal naplók írására. Ezen kívül mérvadó lehet az írás/olvasás gyorsasága, valamint az az időszázalék, melyet a diszk az írási/olvasási műveletekkel tölt, ennek közel 100%-os értéke azt mutatja, hogy ez a szűk keresztmetszet.
- Hálózati kihasználtság: Az interfészekre és teljesítményekre vonatkozó statisztikák protokoll analizátorral támogatva nagyon hasznosak lehetnek a hibaelhárítás szempontjából. Viszont nem szabad csak erre alapozni, hiszen csak pillanatnyi képet kaphatunk a hálózatról, melyet nagymértékben befolyásolhat a felhasználók pillanatnyi viselkedése. Ezért célszerű ezt a vizsgálatot a felhasználók nélkül vagy pedig folyamatosan végezni.
- Magának a tűzfalnak a konfigurálása meglehetősen bonyolult lehet, hogy a teszteléshez optimális beállításokat találhassunk a következőkre különösen érdemes odafigyelni:
- Általános statisztikák a tűzfallal kapcsolatban lévő, különböző hálózati szegmensekről: média hibák, ütközések, broadcastok, maximum átvitel.
- Körbejárási késletetés ideje a tűzfalon áthaladó forgalomnak. Protokoll analízis segítségével meghatározhatjuk, nyomonkövethetjük a kliens-hoszt átvitelt a tűzfallal és tűzfal nélkül is. A kettőt összehasonlítva megnézhetjük az átviteli teljesítmény alakulását. Fellelünk-e egy sokkal nagyobb keretek közötti lyukat a csomagok között, amikor a tűzfalon keresztül mennek az adatok?
- Tűzfal statisztikák. Ha a tűzfal LAN statisztikái jók, akkor megvizsgálhatjuk a következő dolgokat: maximum küldött-kapott csomag/byte egy adott időintervallumon belül. Csomagok száma, melyek sorban állnak a tűzfalnál feldolgozásra várva (átvitel, átviteli hibák, stb.), Ha két hálózati interfésze van a tűzfalnak, akkor

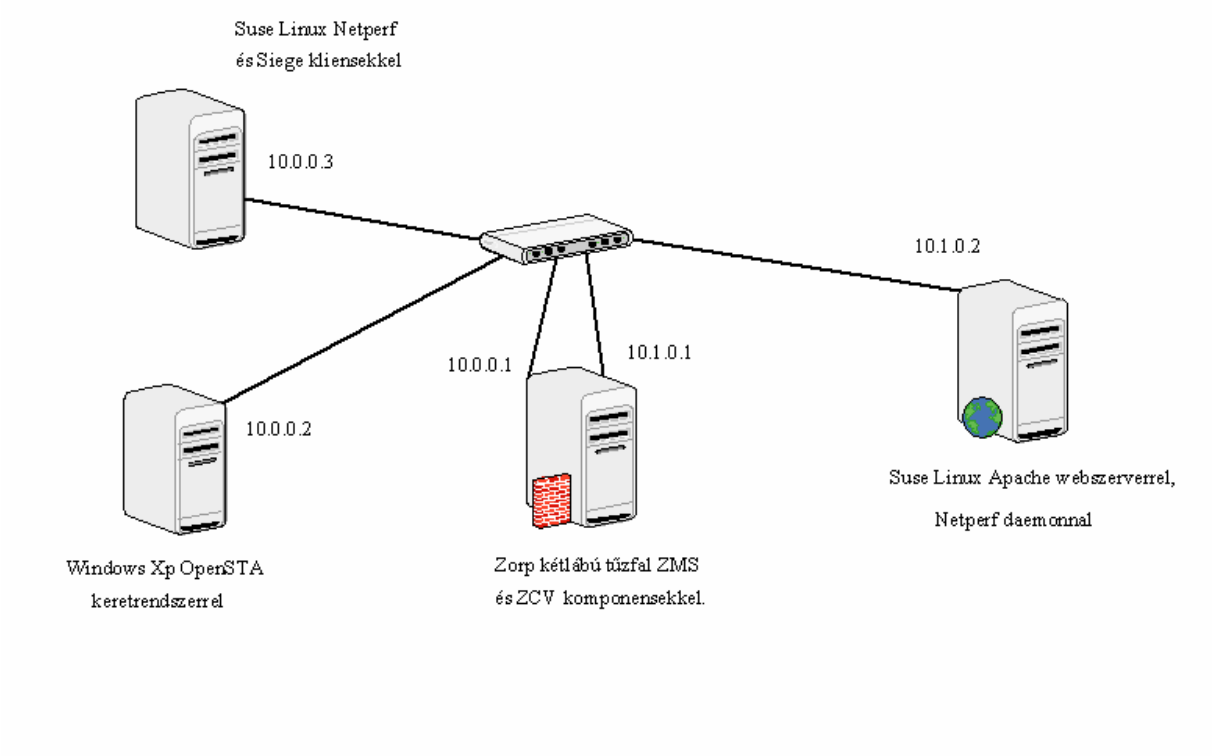
megvizsgálhatjuk, hogyha az egyiken a maximum kapott csomag/másodperc száma korrelál-e a másik interfészen a maximum küldött csomagok/másodperc számával. Ha megfigyelhető késleltetés van a két csúcs között, akkor a tűzfal késleltetést add hozzá az átvitelhez. [12], [18]

4.3. Tesztrendszer

A kialakításra került tesztrendszer egy négy hosztból álló hálózat. A gépek mindegyike egy switch ugyanazon VLAN szegmensében helyezkedik el, két IP cím tartományban, egy belső (intranet), és egy külső (internet) tartományt szimulálva. A topológia így lényegében egy „bastion hoszt" típusú lett. A négy hoszt egy Window XP munkaállomás, egy Linux munkaállomás, melyről a kéréseket generáljuk egy külső címen lévő szerver felé, melyen apache fut, illetve a Zorp tűzfal. A forgalom minden esetben a tűzfalon halad keresztül, mely egy kétlábú tűzfal, a két hálózati kártyán az adott hálózati szegmens IP címei közül kapott egyet. A belső hálózat tartománya 10.0.0.0, netmaskja 255.255.255.0. A külső szegmens tartománya pedig a 10.1.0.0, netmaskja 255.255.255.0. Mindkét esetben az alapértelmezett átjáró a tűzfal hoszt, melynek két hálózati kártyája rendre a 10.0.0.1 és 10.1.0.1 IP címeket kapta. A belső és külső hálózaton lévő gépek emelkedő sorrendben kapják címeiket. Minden gépet IP címeiken keresztül lehet elérni, feleslegesnek tartottam DNS szervert is beállítani.
A következő hardver konfigurációkkal volt a rendszer összeállítva:

- Switch: KTI 24 portos 10/100 SmartSwitch. Típusa: KS-2402
- Windows hoszt: Pentium 4, 2,6 GHz, 1 GB RAM, operációs rendszer: Microsoft XP Service Pack2
- Zorp hoszt: Pentium 4, 26 GHZ, 512 MB RAM, operációs rendszer: ZorpOS 3.1.1.
- Szerver hoszt: Pentium 4 3 GHZ, 1 GB RAM, operációs rendszer: SUSE Linux 10.1 evaluation version, szöveges telepítésű, semmilyen grafikus szolgáltatás nem fut rajta. A kiszolgáló szerver az apache 2.0.54-es verzióját futtatja.
- Linux hoszt: Pentium 4, 3 GHz, 512 RAM memória, operációs rendszer: Suse Linux 10.1 evaluation version, gnome desktop környezet.

A szerver gépnél fontosnak tartottam, hogy a minimális szoftverek legyenek telepítve rá, így minden erőforrását egyetlen feladatának, a beérkező kérések kiszolgálásának szentelhesse.



13. ábra: Teszt környezet topológiája

Szándékom szerint a mérés során három programmal generálom a kéréseket, ezek az OpenSTA (Open System Testing Architecture), a Siege, valamint a Netperf. Míg az első két program HTTP kérések generálására képes, s így kiválóan alkalmasak a Zorpban található HTTP proxy és a ZCV HTTP moduljainak tesztelésére, az utolsó a nyers adatátviteli sebesség mérésére alkalmas, azaz TCP és UDP adatfolyamokat képes előállítani, s ezek paramétereit változtatni. Természetesen a megfelelő konfigurációs beállítások szükségesek ahhoz, hogy mindezen forgalom át is jusson a tűzfalon.

Legelőször az OpenSTA-t mutatom be. Ennek a szoftvernek az 1.4.3-as verziójával dolgoztam. A szoftver maga egy keretrendszer, mely nyílt forráskódú és a GNU GPL alatt terjeszthető. A CORBA architektúrára épül s elosztott teljesítménymérést tesz lehetővé. A szoftver Windows alapú. Különböző felhasználói szkripteket futtat le, melyek a felhasználói viselkedést szimulálják, így képes lehetne meglehetősen dinamikus viselkedési formák előállítására. Az eredményeket és a statisztikákat különböző automatikus és felhasználó által kontrollált mechanizmusokon keresztül gyűjti (időzítők, SNMP adatok, Windows teljesítmény értékek, HTTP késleltetések). Az adatok már mérés során, valós időben rendelkezésre állnak, legvégül pedig naplókba kerülnek, melyek tetszőlegesen filterezhetőek. A szoftver a HTTP kérések, és a kiszolgálás közötti kapcsolatokat térképezi fel, ezért igen alkalmas a tűzfal késletetés-változásának vizsgálatára. A következő alapfunkciókat valósítja

meg, HTTP átvitel (bájt/másodperc), HTTP válasz idő a válaszok számának függvényében, HTTP hibák és a HTTP kérések arányának számolása, stb. Sajnos a program nem váltotta be a hozzá fűzött reményeket, meglehetős instabilitást mutatott, sok érthetetlen kapcsolatbontási és időtúllépési esetet produkálva, melyek a Siege-gel folytatott hasonló méréseknél nem jelentkeztek. (A tűzfal kiiktatásával végezz mérésnél szinte nem is fordultak elő időtúllépéses hibák, csak amikor a Zorp-on keresztül haladtak keresztül a csomagok, némi irodalom keresés után úgy tűnt, ez nem egyedi probléma). Valószínűleg a keretrendszer inkompatibilitása okozza a hibát, s nem a tűzfal sajátos csomagszűrési beállításai. A szoftver legnagyobb előnye az elosztott terhelés generálása lett volna, amiről így kénytelen voltam lemondani. [20]

A második tesztprogram a Linux platformon (AIX, Solaris, BSD. HP-UX rendszerekre portolható) futó Siege 2.64-es verziója volt. A szoftver háromféle működési módot ismer, egy fájlból kiolvassa az URL-ek listáját, s ezekre küld kéréseket, inkrementális vagy véletlen módon bejárva ezt a halmazt, vagy pedig egyetlen URL-re küld kérdéseket. A teszt során én ez utóbbi viselkedési módot használom majd. Képes meglehetősen nagyszámú felhasználót szimulálni (1010), illetve megadható, hogy ezen felhasználók között mekkora késleltetés legyen (minél kisebb, annál jobban megterheli a szervert vele). Beállítható a futás hosszúsága tranzakció szám és idő szerint is Ezek közül az utóbbit használtam. Különböző statisztikák készítésére képes, ezek közül a fontosabbak:

- Tranzakciók száma: azaz az adott url-t hányszor próbálta letölteni.
- Teszt teljes ideje
- Rendelkezésre állás: hányszor tudta letölteni a fájlt, s hányszor szakadt meg a kapcsolat (itt a 400-on felüli HTTP üzenetek nem jelenek meg a tranzakciók között)
- Kliensenkénti átvitt adat: az az adat, melyet az egyes kliensek letöltöttek, ebben a különböző fejléc információk is benne vannak, ezért nagyobb lehet, mint a szerveren tárolt fájl mérete, akár fájlonként is változhat ez a méret.
- Válasz idő: a kliensek kéréseire adott átlagos válasz idő
- Tranzakciók száma másodpercenként (tranzakciók száma osztva az eltelt idővel).
- Átviteli sávszélesség: a másodpercenkénti átvitt bitek száma az összes szimulált felhasználóra tekintve.
- Párhuzamosság: a párhuzamos kapcsolatok átlagos száma (minél több, annál megterhelőbb a szerver számára)
- Sikeres tranzakciók száma (azon válaszok száma, melyek kódja kisebb, mint 400)

A TCP átvitel tesztelésére felhasznált szoftver a netperf 2.4.1-es verziója volt, mely TCP és UDP csomagok generálására és mérésére szolgál. Kliens-szerver alapú program, s a két gép közötti adatátvitelt méri le. Ezek közül a legfontosabb talán az átviteli gyorsaság, illetve a különböző válasz idők gyakoriságának jelzése. Bár képes IPv4 és IPv6 kezelésére is, a teszt során csak az IPv4-et használtam, mivel túl nyomó részben még mindig ezt használják a legtöbb helyen és hálózatban. A program először egy vezérlő TCP kapcsolatot nyit a távoli rendszer felé. Ezen a kapcsolaton keresztül történik a teszt konfigurációjának s egyéb utasításainak átvitele. Ezután egy külön adatátviteli csatorna jön létre, mely a teszt után le is bontódik. Legvégül a vezérlőkapcsolaton keresztül visszaküldődnek az adatok. A teszt során a vezérlőcsatornán nincsen forgalom. Lehetőség van különböző puffer méreteket beállítani a forrás és célrendszerben, de miután néhány ellenőrző mérést végeztem, nem találtam nagy különbséget a mért adatok között, ezért hagytam az alapértelmezett beállításokon működni. A program lehetőséget ad saját tesztszkriptek használatára, én azonban a beépítetteket használtam, melyek már eleve megfeleltek a mérési céloknak. Ezek a tesztek átviteli és kérés-válasz késletetési tesztekre különíthetőek el:

Fontosabb átviteli tesztek:

- TCP_STREAM: Az alapértelmezett átviteli teszt a programban. Míg a kapcsolatépítési idő nincs benne a tesztben, addig az utolsó küldött adatcsomagok megérkezése igen.
- TCP_MAERTS: Ugyanaz a teszt, csak az adatátvitel az ellenkező irányban történik. Ez lényegében a mérések ellenőrzésére szolgálhat.

Kérés-válasz késletetési tesztek:

- TCP_RR: A tranzakciós arányt méri, melyet az összes sikeres tranzakció és a teljes átviteli idő hányadosaként kaphatunk meg. A kapcsolatkiépítési idő nem számít bele a teljes átviteli időbe.
- TCP_CC: Ez a teszt az előzőnek a kiegészítése, csak a TCP kapcsolatkiépítési és lebontási idejének mérésére szolgál, viszont semmilyen kérés és válasz nem küldődik át a hálózaton.

[21]

Az egyes hosztok erőforrás kihasználtságának mérésére a vmstat 3.2.1-es verzióját használtam. Mivel előzetes mérések alapján kiderült, hogy a szűk keresztmetszet a tűzfal, ezért a későbbiekben sem az apache szerveren, sem a kéréseket generáló hosztokon nem futtattam a programot. Statisztikát készít a futó a futó folyamatokról, a mérés szempontjából fontosak:

- Futásra váró processzek száma (r)
- Erőforrásra (I/O, memórialap, stb.) váró processzek száma (b)
- A használt virtuális memória nagysága (swpd)
- A szabad memória mérete(free)
- A pufferként használt memória mérete (buff)
- A cachelésre használt memória (cache)
- A diszkre irt virtuális memória nagysága (si)
- A diszkről olvasott memórianagysága (so)
- A másodpercenkénti megszakítások száma (in)
- A másodpercenkénti környezetváltások száma (cs)
- A CPU által nem kernel módban futó kód ideje
- A CPU által kernel módban futtatott kód ideje
- A CPU által várakozással telt idő (id)
- Az IO-ra várt idő (wa)

Ha a futási sorban (procs r) folyamatosan több processz van, mint ahány cpu a rendszerben, az kisebb lassulás oka lehet

Ha ez a szám huzamos ideig több, mint a processzorok számának négyszerese, már komoly processzor-erőforrás hiánnyal van dolgunk

Ha a szabad processzor idő (cpu id) folyamatosan 0, és a system módban töltött idő duplája, vagy több, mint a user módban töltött idő, szintén CPU erőforrás hiánnyal állunk szemben[22]

4.4. Mérési jegyzőkönyv

A tesztelés során több teszt fájlt is generáltam, amint már elhangzott, ezek mérete a 7430 bájttól 1791170 bájtig terjedt így a teszteseteket a különböző modul beállítások, valamint a felhasználók száma befolyásolta. Legelőször a netperf mérést végeztem el, minden mérés 20 másodpercig tartott. A felhasznált parancs:

netperf –t *tesztnév* –H *hosztnév* –l 20 –f K

Teszt típusa	Eredményei
TCP STREAM	9609,55 Kbájt/mp
TCP MAERTS	8845.50 Kbájt/mp
TCP RR	5698.28 tranzakció/mp
TCP CC	559.15 tranzakció/mp

1. Táblázat: netperf eredmények

A következőkben a siege programot futtattam, melyre a teszt fájlok a következőek voltak:

Filenév	Méret (bájt)
kicsi.htm	7430
kozepes.htm	48699
nagyobb.html	118675
500k.html	459098
netperf-2.4.1.tar.gz	1519266
orias.html	1791170

2. Táblázat: Méréshez használt fájlok

Minden html fájl tartalmazott javascript kódot, illetve a HTML modul kiszűrte a title tag-et az oldalból. Így a modul funkcionalitását is ellenőrizhettük, valamint biztosak lehettünk abban, hogy végez is szűrést, s ez von el erőforrásokat.
A teszteket minden esetben 60 másodpercig futtattam, ez a gyakorlatban azt jelentette, hogy a szervert 1 percig 1000, 500 vagy 250 felhasználó folyamatosan lekérdezésekkel bombázta. Minden tesztet többször megismételtem, s az eredmények átlagát vettem. A következő paranccsal történt mindez:

siege –t60s -c*felhasználószám hosztnév*

Előzetes tesztmérésekkel megállapítottam, hogy a folyamatos terhelés miatt (főleg a HTML modul használata esetén) nagyon sok időtúllépéses kapcsolatbontás jön létre. Ezt ellensúlyozandó a kliens oldali várakozási időt 60 másodpercre állítottam, ami valamelyest javított az arányon, a teszteredményeken jól látszik, hogy sok esetben kihasználták ezt a kliensek.

Kizárólag csomagszűrést használó konfiguráció teszteredményei

kicsi.htm			
Felhasználók száma	1000	500	250
Tranzakciók száma	69423	70720	71900
Rendelkezésre állás %	99,60%	100,00%	100,00%
Mérés ideje mp	60,2	59,67	60,47
Adatátvitel Mb	497,41	506,71	515,16
Válasz idő mp	0,8	0,4	0,2
Tranzakciók gyorsasága tr/mp	1163,45	1185,19	1189,2
Átviteli sávszélesség Mb/mp	8,34	8,49	8,52
Párhuzamosság aránya	69423	479,41	241,95
Sikeres tranzakció	69423	70720	71900
Nem sikerült tranzakciók	26	0	0
Leghosszabb tranzakció	47,23	39,04	20,98

3. Táblázat: kicsi.hm eredményei csomagszűrés esetén

nagyobb.html			
Felhasználók száma	1000	500	250
Tranzakciók száma	4773	5245	5335
Rendelkezésre állás %	91,82%	97,75%	100,00%
Mérés ideje mp	60,23	59,7	60,46
Adatátvitel Mb	550,57	605,01	615,4
Válasz idő mp	5,79	4,19	2,76
Tranzakciók gyorsasága tr/mp	79,25	87,86	88,24
Átviteli sávszélesség Mb/mp	9,14	10,13	10,18
Párhuzamosság aránya	458,45	367,75	243,72
Sikeres tranzakció	4773	5245	5335
Nem sikerült tranzakciók	425	121	0
Leghosszabb tranzakció	56,65	47,29	23,79

4. Táblázat: nagyobb.html eredményei csomagszűrés esetén

orias.html			
Felhasználók száma	1000	500	250
Tranzakciók száma	297	287	307
Rendelkezésre állás %	25,69%	57,98%	84,11%
Mérés ideje mp	60,48	60,32	59,63
Adatátvitel Mb	507,33	490,25	524,42
Válasz idő mp	32,96	33,27	26,93
Tranzakciók gyorsasága tr/mp	4,91	4,76	5,14
Átviteli sávszélesség Mb/mp	8,39	8,13	8,78
Párhuzamosság aránya	161,86	158,28	138,4
Sikeres tranzakció	297	287	307
Nem sikerült tranzakciók	859	208	58
Leghosszabb tranzakció	80,16	55,54	54,88

5. Táblázat: orias.html eredményei csomagszűrés esetén

A plug proxyn keresztül haladó HTTP forgalom mérésének eredményei

kicsi.htm			
Felhasználók száma	1000	500	250
Tranzakciók száma	14050	14069	14248
Rendelkezésre állás %	97,32%	98,16%	99,08%
Mérés ideje mp	60,22	59,62	60,07
Adatátvitel Mb	100,67	100,8	102,09
Válasz idő mp	1,25	0,76	0,53
Tranzakciók gyorsasága tr/mp	233,31	235,98	237,19
Átviteli sávszélesség Mb/mp	1,67	1,69	1,7
Párhuzamosság aránya	290,92	178,92	125,85
Sikeres tranzakció	14050	14069	14248
Nem sikerült tranzakciók	387	263	133
Leghosszabb tranzakció	58,63	48,53	48,45

6. Táblázat: kicsi.htm eredményei Plug proxy esetén

kozepes.htm			
Felhasználók száma	1000	500	250
Tranzakciók száma	8826	8715	8858
Rendelkezésre állás %	95,97%	97,17%	98,82%
Mérés ideje mp	60,11	59,89	60,22
Adatátvitel Mb	413,86	408,66	415,36
Válasz idő mp	2,92	1,66	0,99
Tranzakciók gyorsasága tr/mp	146,83	145,22	147,22
Átviteli sávszélesség Mb/mp	6,89	6,82	6,9
Párhuzamosság aránya	428,16	242,19	146,16
Sikeres tranzakció	8826	8715	8858
Nem sikerült tranzakciók	371	254	106
Leghosszabb tranzakció	58,66	45,68	48,96

7. Táblázat: kozepes.htm eredményei Plug proxy esetén

nagyobb.html			
Felhasználók száma	1000	500	250
Tranzakciók száma	4557	4349	4350
Rendelkezésre állás %	90,81%	95,67%	97,36%
Mérés ideje mp	60,34	60,48	59,66
Adatátvitel Mb	525,66	501,66	501,78
Válasz idő mp	4,31	3,04	1,84
Tranzakciók gyorsasága tr/mp	75,52	71.92	72,91
Átviteli sávszélesség Mb/mp	8,71	8,29	8,41
Párhuzamosság aránya	325,54	218,6	133,94
Sikeres tranzakció	4557	4349	4350
Nem sikerült tranzakciók	461	197	118
Leghosszabb tranzakció	51,96	52,54	49,08

8. Táblázat: nagyobb.html eredményei Plug proxy esetén

500k.html			
Felhasználók száma	1000	500	250
Tranzakciók száma	1373	1362	1387
Rendelkezésre állás %	76,11%	81,31%	90,59%
Mérés ideje mp	60,37	59,93	59,66
Adatátvitel Mb	601,14	596,32	607,27
Válasz idő mp	5,87	5,2	4,33
Tranzakciók gyorsasága tr/mp	22,74	22,73	23,25
Átviteli sávszélesség Mb/mp	9,96	9,95	10,18
Párhuzamosság aránya	133,6	118,08	100,73
Sikeres tranzakció	133,6	1362	1387
Nem sikerült tranzakciók	1373	313	144
Leghosszabb tranzakció	54,61	59,27	57,07

9. Táblázat: 500k.html eredményei Plug proxy esetén

netperf-2.4.1.tar.gz			
Felhasználók száma	1000	500	250
Tranzakciók száma	379	357	368
Rendelkezésre állás %	59,03%	70,04%	66,49%
Mérés ideje mp	60,37	60,06	59,77
Adatátvitel Mb	549,13	562,17	546,23
Válasz idő mp	15,6	12,6	11
Tranzakciók gyorsasága tr/mp	6,28	6,46	6,31
Átviteli sávszélesség Mb/mp	9,1	9,36	9,14
Párhuzamosság aránya	97,97	81,39	69,41
Sikeres tranzakció	379	388	377
Nem sikerült tranzakciók	263	166	190
Leghosszabb tranzakció	59,99	59,36	48,69

10. Táblázat: netperf-2.4.1.tar.gz eredményei Plug proxy esetén

orias.html			
Felhasználók száma	1000	500	250
Tranzakciók száma	315	311	317
Rendelkezésre állás %	50,24%	49,69%	61,20%
Mérés ideje mp	60,14	60,16	59,62
Adatátvitel Mb	538,08	531,25	541,5
Válasz idő mp	18,39	16,36	11,02
Tranzakciók gyorsasága tr/mp	5,24	5,17	5,32
Átviteli sávszélesség Mb/mp	8,95	8,83	9,08
Párhuzamosság aránya	96,33	84,58	58,58
Sikeres tranzakció	315	311	317
Nem sikerült tranzakciók	312	316	201
Leghosszabb tranzakció	60,02	59,87	47,35

11. Táblázat: orias.html eredményei Plug proxy esetén

HTTP proxy teszteredményei

kicsi.htm			
Felhasználók száma	1000	500	250
Tranzakciók száma	13295	13289	13262
Rendelkezésre állás %	96,03%	98,73%	99,17%
Mérés ideje mp	60,1	60,16	59,89
Adatátvitel Mb	95,26	95,22	95,03
Válasz idő mp	1,69	0,9	0,67
Tranzakciók gyorsasága tr/mp	221,21	220,89	221,44
Átviteli sávszélesség Mb/mp	1,58	1,58	1,59
Párhuzamosság aránya	373,63	199,17	148,1
Sikeres tranzakció	13295	13289	13262
Nem sikerült tranzakciók	550	171	111
Leghosszabb tranzakció	58,38	58,63	36,96

12. Táblázat: kicsi.htm eredményei HTTP proxy esetén

kozepes.htm 250-es és 1000es csere			
Felhasználók száma	1000	500	250
Tranzakciók száma	8545	8511	8423
Rendelkezésre állás %	97,39%	97,16%	99,08%
Mérés ideje mp	60,55	59,87	60,1
Adatátvitel Mb	400,69	399,09	394,96
Válasz idő mp	2,27	1,75	1,24
Tranzakciók gyorsasága tr/mp	141,12	142,16	140,15
Átviteli sávszélesség Mb/mp	6,62	6,67	6,57
Párhuzamosság aránya	320,53	248,31	174,31
Sikeres tranzakció	8545	8511	8423
Nem sikerült tranzakciók	229	249	78
Leghosszabb tranzakció	59,1	49,01	48,03

13. Táblázat: kozepes.htm eredményei HTTP proxy esetén

nagyobb.html			
Felhasználók száma	1000	500	250
Tranzakciók száma	4479	4468	4418
Rendelkezésre állás %	97,71%	94,46%	98,48%
Mérés ideje mp	59,64	59,57	59,56
Adatátvitel Mb	516,66	515,39	509,62
Válasz idő mp	3,13	3,03	2,22
Tranzakciók gyorsasága tr/mp	75,1	75	74,18
Átviteli sávszélesség Mb/mp	8,66	8,65	8,56
Párhuzamosság aránya	235,23	226,98	164,56
Sikeres tranzakció	4479	4468	4418
Nem sikerült tranzakciók	105	262	68
Leghosszabb tranzakció	59	49,05	59,01

14. Táblázat: nagyobb.html eredményei HTTP proxy esetén

500k.html			
Felhasználók száma	1000	500	250
Tranzakciók száma	1341	1373	1398
Rendelkezésre állás %	79,21%	84,42%	90,19%
Mérés ideje mp	60,13	59,55	60,23
Adatátvitel Mb	587,13	595,45	612,09
Válasz idő mp	8,02	5,67	4,74
Tranzakciók gyorsasága tr/mp	22,3	22,84	23,21
Átviteli sávszélesség Mb/mp	9,76	10	10,16
Párhuzamosság aránya	178,87	129,55	109,97
Sikeres tranzakció	1341	1360	1398
Nem sikerült tranzakciók	352	251	152
Leghosszabb tranzakció	59,81	58,21	40,78

15. Táblázat: 500k.html eredményei HTTP proxy esetén

netperf-2.4.1.tar.gz			
Felhasználók száma	1000	500	250
Tranzakciók száma	359	357	368
Rendelkezésre állás %	44,76%	50,21%	72,44%
Mérés ideje mp	60,12	60,33	60,45
Adatátvitel Mb	520,15	517,25	533,19
Válasz idő mp	26,76	22,63	17,9
Tranzakciók gyorsasága tr/mp	5,97	5,92	6,09
Átviteli sávszélesség Mb/mp	8,65	8,57	8,82
Párhuzamosság aránya	159,81	133,89	108,95
Sikeres tranzakció	359	357	368
Nem sikerült tranzakciók	443	354	140
Leghosszabb tranzakció	58,9	60,16	59,5

16. Táblázat: netperf-2.4.1.tar.gz eredményei HTTP proxy esetén

orias.html			
Felhasználók száma	1000	500	250
Tranzakciók száma	294	293	297
Rendelkezésre állás %	47,28%	50,69%	59,40%
Mérés ideje mp	59,74	60,35	60,37
Adatátvitel Mb	505,63	500,5	507,33
Válasz idő mp	24,18	26,86	15,31
Tranzakciók gyorsasága tr/mp	4,95	4,86	4,92
Átviteli sávszélesség Mb/mp	8,46	8,29	8,4
Párhuzamosság aránya	119,8	130,4	75,3
Sikeres tranzakció	296	293	297
Nem sikerült tranzakciók	330	285	203
Leghosszabb tranzakció	59,48	58,45	59,01

17. Táblázat: orias.html eredményei HTTP proxy esetén

HTTP proxy és HTML modul szűrési eredményei

kicsi.htm			
Felhasználók száma	1000	500	250
Tranzakciók száma	7623	7490	7089
Rendelkezésre állás %	92,09%	96,21%	98,54%
Mérés ideje mp	59,97	59,72	60,27
Adatátvitel Mb	13,64	13,56	13,13
Válasz idő mp	2,35	1,7	1,29
Tranzakciók gyorsasága tr/mp	127,11	125,42	117,62
Átviteli sávszélesség Mb/mp	0,23	0,23	0,22
Párhuzamosság aránya	299,35	212,67	151,5
Sikeres tranzakció	3215	3508	3936
Nem sikerült tranzakciók	655	295	105
Leghosszabb tranzakció	47,15	47,84	45,2

18. Táblázat: kicsi.htm eredményei HTTP proxy és HTML modul esetén

kozepes.htm			
Felhasználók száma	1000	500	250
Tranzakciók száma	3932	3855	3729
Rendelkezésre állás %	88,16%	94,94%	97,72%
Mérés ideje mp	60,23	59,59	59,64
Adatátvitel Mb	41,95	40,6	42,76
Válasz idő mp	3,18	1,92	1,66
Tranzakciók gyorsasága tr/mp	65,28	64,69	62,53
Átviteli sávszélesség Mb/mp	0,7	0,68	0,72
Párhuzamosság aránya	207,74	124,13	103,57
Sikeres tranzakció	1521	1502	1562
Nem sikerült tranzakciók	528	197	87
Leghosszabb tranzakció	58,13	58,37	58,38

19. Táblázat: kozepes.htm eredményei HTTP proxy és HTML modul esetén

nagyobb.html			
Felhasználók száma	1000	500	250
Tranzakciók száma	2696	2589	2584
Rendelkezésre állás %	85,18%	94,39%	97,36%
Mérés ideje mp	59,74	60	60,17
Adatátvitel Mb	65,79	68,44	67,65
Válasz idő mp	3,09	2,47	2,08
Tranzakciók gyorsasága tr/mp	45,13	43,15	42,94
Átviteli sávszélesség Mb/mp	1,1	1,14	1,12
Párhuzamosság aránya	139,28	106,78	89,38
Sikeres tranzakció	824	859	851
Nem sikerült tranzakciók	469	154	70
Leghosszabb tranzakció	48,24	58,94	58,84

20. Táblázat: nagyobb.html eredményei HTTP proxy és HTML modul esetén

500k.html			
Felhasználók száma	1000	500	250
Tranzakciók száma	2481	2472	2361
Rendelkezésre állás %	84,47%	95,78%	94,06%
Mérés ideje mp	60,37	60,48	60,1
Adatátvitel Mb	61,65	62,88	60,65
Válasz idő mp	3,05	2,61	1,97
Tranzakciók gyorsasága tr/mp	41,1	40,87	39,28
Átviteli sávszélesség Mb/mp	1,02	1,04	1,01
Párhuzamosság aránya	129,38	106,53	77,48
Sikeres tranzakció	618	603	625
Nem sikerült tranzakciók	456	109	149
Leghosszabb tranzakció	58,62	58,65	49,57

21. Táblázat: 500k.html eredményei HTTP proxy és HTML modul esetén

netperf-2.4.1.tar.gz			
Felhasználók száma	1000	500	250
Tranzakciók száma	2747	2549	2492
Rendelkezésre állás %	93,06%	96,15%	96,51%
Mérés ideje mp	60,48	59,81	60,37
Adatátvitel Mb	140,44	139,99	143,67
Válasz idő mp	3,04	2,18	2,22
Tranzakciók gyorsasága tr/mp	45,42	42,62	41,28
Átviteli sávszélesség Mb/mp	2,32	2,34	2,38
Párhuzamosság aránya	138,14	92,77	91,43
Sikeres tranzakció	743	767	783
Nem sikerült tranzakciók	205	102	90
Leghosszabb tranzakció	58,98	51,7	59,31

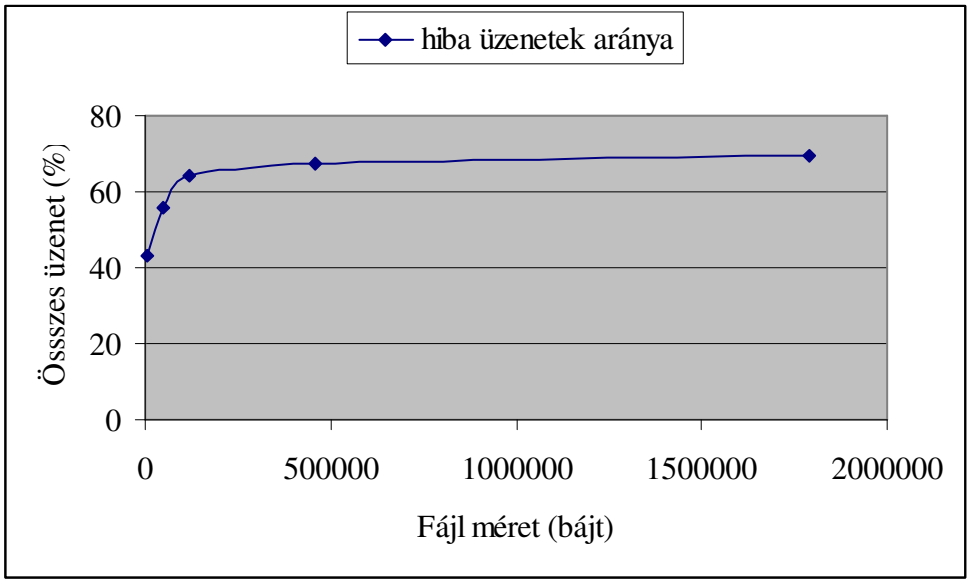
22. Táblázat: netperf-2.4.1.tar.gz eredményei HTTP proxy és HTML modul esetén

orias.html			
Felhasználók száma	1000	500	250
Tranzakciók száma	2472	2468	2237
Rendelkezésre állás %	89,21%	96,56%	96,42%
Mérés ideje (mp)	60,05	59,65	60,06
Adatátvitel Mb	59,55	59,38	63,39
Válasz idő (mp)	2,66	2,25	2,07
Tranzakciók gyorsasága tr/mp	41,17	41,37	37,25
Átviteli sávszélesség Mb/mp	0,99	1	1,06
Párhuzamosság aránya	109,68	93,28	77,14
Sikeres tranzakció	623	621	604
Nem sikerült tranzakciók	299	88	83
Leghosszabb tranzakció (mp)	52,46	59,51	59,3

4.5. Eredmények értékelése

A legelső mérésekből kiderült, hogy mekkora lenne a tűzfal teljes áteresztőképessége, mintegy 600 Mb környéki áttöltési mennyiséget tud produkálni a tűzfal 60 másodperc alatt. Ez megfelel egy 100 Mbites hálózati kártya átviteli képességének Azonban érdekes, hogy ezt nem csak a minimális csomagszűrési szabályok esetén képes produkálni a tűzfal, még a HTTP proxy használatával is megközelíti ezt a mennyiséget. Azonban nagyon befolyásolja ezt a fájl mérete, míg a legkisebb fájloknál a CPU kihasználtsága a szűk keresztmetszett, révén, hogy egyszerűen nem képes több kapcsolatot és szálat nyitni a tűzfal. Itt el kell különítenünk egymástól a három féle tesztet. A tartalomszűrési tesztnél egyértelműen a CPU a legszűkebb erőforrás. Minden fájl méretnél és felhasználó számnál 100%-os processzor használat jelenik meg. Így nagyon sokszor jön létre időtúllépés miatt félbeszakított tranzakció, de gyakran előfordul az is, hogy a socket hibássá válik (socket read error). Ráadásul gyakran történik meg, hogy mivel nem képes elég időt foglalkozni az adatfolyammal, 500-as hibát ad vissza a kívánt oldal helyett, azaz elvileg a kérést hibás protokoll üzenetnek értelmezi. Mivel azonban ezek a kérések hagyományos HTTP proxyn keresztül hibátlanul visszaadják a kért fájlokat, ezért valószínűsíthető, hogy másról van szó. A zcv.conf fájlban található várakozási utasítás értékének változtatása hatással van ezen 500-as hibaüzenetet generálódásának gyakoriságára. Tehát ennek a problémának érdemes lehet utánajárni az implementáció szemszögéből is. Az egyes fájlok letöltésekor nagyon változó a teljes adatmennyiség (sok ismétlési kérés lehet), így ez is azt mutatja, hogy nehezen szolgál ki ennyi kérést a tűzfal. Információtartalommal rendelkezik az is, hogy viszonylag alacsonyszintű kérés generálás esetén nem jön elő ez a

hiba. Itt nagyon érdekes lehet megnézni az összes tranzakció, a sikeres tranzakciók és a nem sikerült tranzakciók arányainak összehasonlítását. Ha a legkisebb fájlméretet vesszük is és a legkevesebb felhasználót, akkor is 7089-ből mindössze 3936 sikerült és 105 pedig megszakadt valamiféle módon. Ez azt jelenti, hogy 3048 letöltés végződött 500-as hibaüzeneteben! Minél feljebb visszük a fájl méretet, annál jobban eltolódik ez az arány a hibás protokoll kérést jelző üzenetek felé. Nem csak a stream modulban fordultak elő, ilyen hibák, hanem a fájl moduloknál is, azonban azt nem mértem ilyen részletesen. A következő kép 250 felhasználó által indított kérés sorozat esetére ábrázolja a hibaüzenetek arányát.



14. ábra: Kérések hibázási aránya

Ez a legkevésbé terhelt összeállítás és mégis sok hibát jelző visszatérési adat van a rendszerben.

Az is észrevehető, hogy az átvitt adat mennyisége nem változik különösebben egy fájl méreten belül a felhasználók számának változtatásával sem. Valamint a fájl méret növekedésével csökken a párhuzamos műveletek és tranzakciók száma, ami 100%-os processzor kihasználtság mellett nem is csoda.

Az adatátvitel változásainál fontos azonban tudnunk, hogy ez azért ilyen kicsi, mert a forgalom nagy részét a hibaüzenetek teszik ki. Érdekes látni, hogy nem html típusú fájl esetén (amit szintén megvizsgál, csak nem végez rajta szűrést) az átvitel is sokkal nagyobb, a 1519266 bájt méretű zip fájl esetén ez 140 Mb körüli érték.

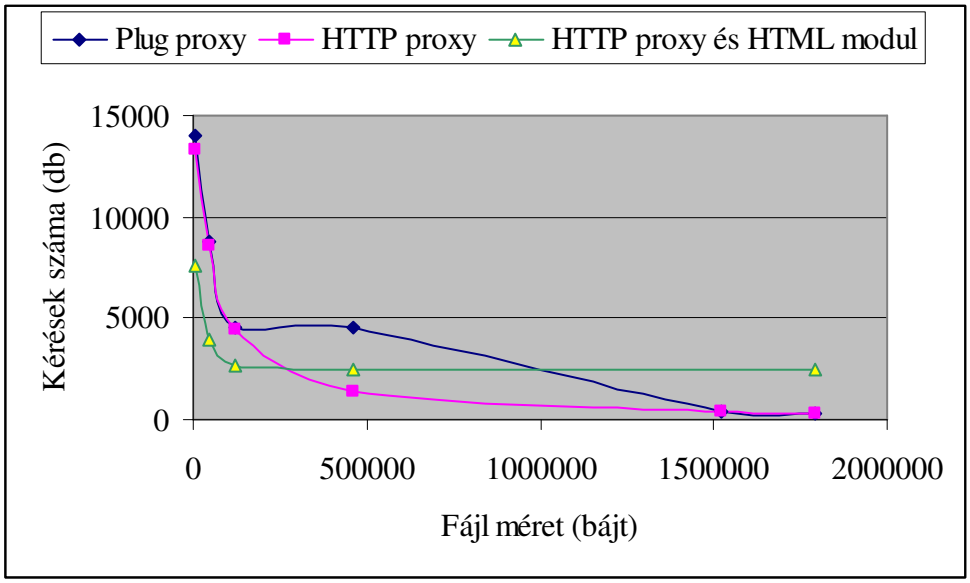
A Plug és a HTTP proxy már nem mutat ilyen működési anomáliákat. Egyetlen egyszer sem tudtam produkálni az előző jelenséget, sőt az azonos méretű fájlok letöltése ugyanolyan adatforgalmat generált minden esetben (tehát például a 7430 bájt méretű fájl letöltéséhez szükséges adatforgalom minden esetben 7513 bájt volt).

Itt valamivel erősebben jön elő a felhasználók számától való függés. De ez csak a párhuzamosság tekintetében és a másodpercenkénti tranzakciók számában nyilvánul meg, mely kevesebb felhasználónál kevesebb, de a teljes adatátvitel nem sokat változik. Ezen kívül a fájlok méretének növekedésével is csökken a párhuzamosítási arány, valamint a másodpercenkénti tranzakciók száma.

A processzor kihasználtsága a fájlméret növekedésével csökkenni kezd, és fokozatosan a hálózat válik a szűk keresztmetszetté. Ezt a használt fájlok közül az 500 Kbájtos méretnél éri el, majd később a fájlméret növekedésével csökkenni kezd a hálózat és a CPU kihasználtsága, viszont igazán sok megszakítás jelentkezik a rendszerben. Mivel látszólag se a memória nem fogy el, se a processzor nem pörög 100%-os kihasználtságon, és a hálózat sincsen annyira leterhelve valószínűleg a fájl méretéből adódik az átvitel csökkenése. Ezen kívül az is feltűnő, hogy a processzor viszonylag sok időt tölt el kernel módban, s mivel ezek a folyamatok nem annyira megszakíthatóak, ez lehet az oka a kisebb adatátviteli mennyiségnek. Az ok valószínűleg a hálókártyában, vagy annak meghajtó programjában leledzik.

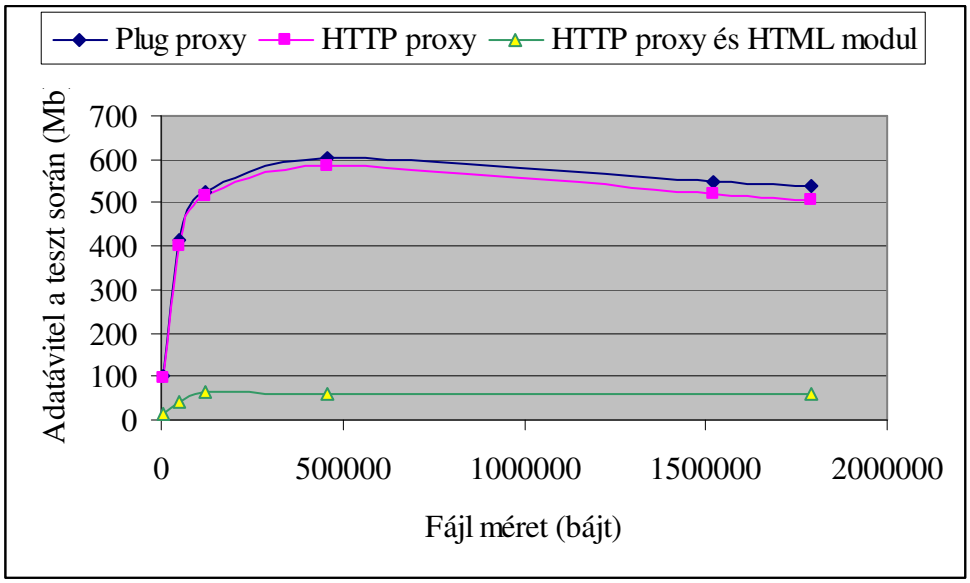
A méréseknél megpróbáltam a stack limit változtatásával élni, azonban ez nem változtatott sokat a kapott eredményeken.

A következő ábrán az 1000 felhasználóra tekintett kérések számát hasonlíthatjuk össze a három esetre. Megtévesztő lehet, hogy a HTML modult használó méréskor ennyivel több kérés generálódik, de ne felejtsük el, ekkor a hibás protokoll üzenetek sokasága miatt válik lehetővé ez a sok kérés. Az azonban jól látható, hogy a HTTP és Plug proxy között nincsen sok különbség.



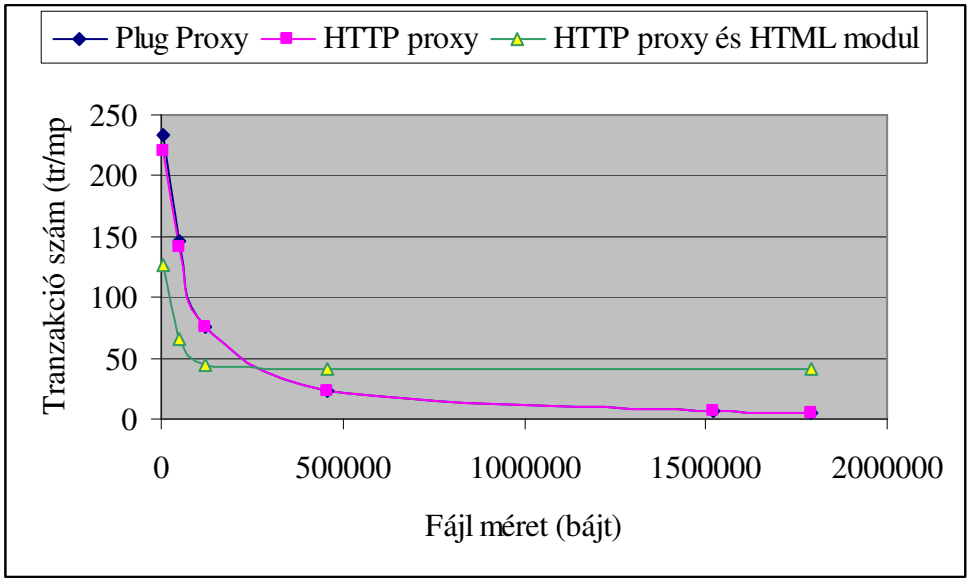
15. ábra: Proxyk és modulok által generált kérések aránya

A következő ábrán az adatátvitel mennyiségét láthatjuk, itt már jól kijön, hogy mennyivel magasabb az adatátvitel a tartalomszűrést nem használó proxyk esetén.



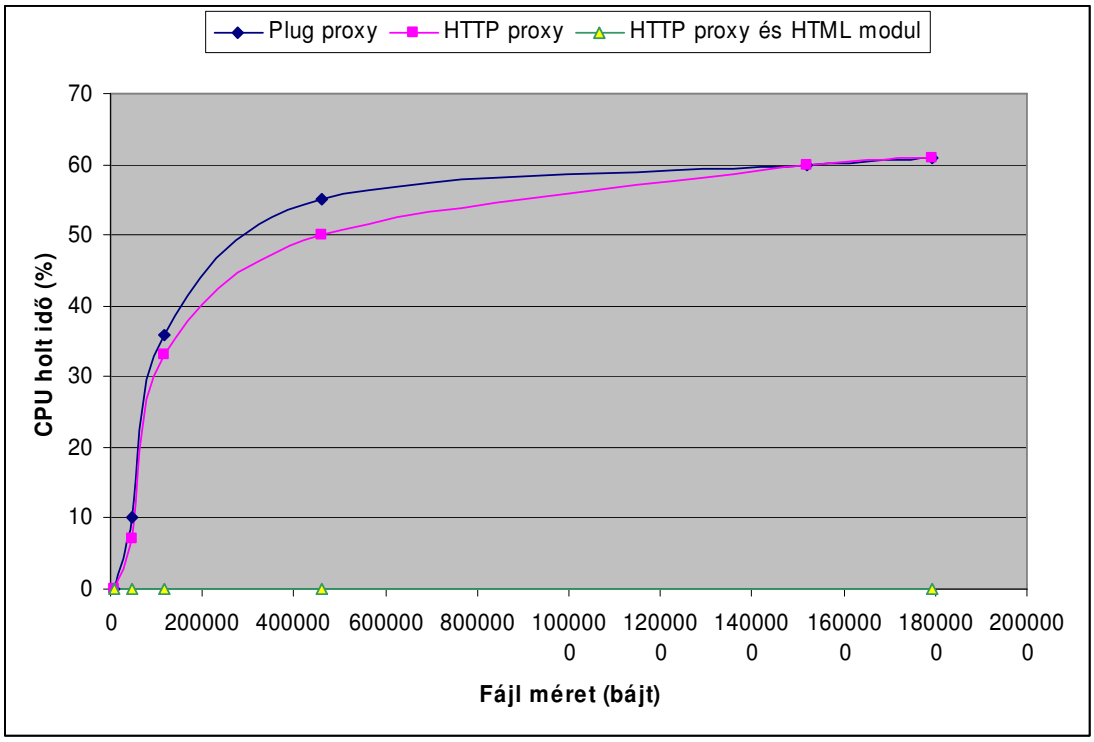
16. ábra: Proxyk és modulok teljes átviteli adatmennyisége

A következő diagrammon az 1000 felhasználó esetén vett másodpercenkénti tranzakció számot láthatjuk, itt szintén csalóka a HTML modul vezetése a másik két proxy előtt, itt szintén a hibaüzenetekből származó tranzakciók miatt tűnik nagyobbnak. A Plug és a HTTP proxy viszont szinte ugyanazt a teljesítményt nyújtja, ami azért kicsit meglepő. Hiszen azért az egyik mégis csak végez mélyprotokoll elemzést.



17. ábra: Proxyk és modulok által végrehajtott tranzakciók száma

A legutolsó ábrán pedig a CPU kihasználtságának arányait láthatjuk, 1000 felhasználóra tekintve.



18. ábra: Proxyk és modulok CPU kihasználása

Végezetül azért szeretném arra felhívni a figyelmet, hogy kicsi adatátvitelt produkált a tűzfal a HTML modul alkalmazásakor ugyan, azonban az általa végzett sikeres tranzakciók

közelítenek a HTTP proxy teljesítményéhez nagyobb fájlok esetén. HTTP proxy használata esetén képes a teljes hálózati sávszélességet kihasználni megfelelő fájlméret esetén, a legfontosabb, sehol nem ütközött memória korlátba.

4.6. Továbbfejlesztési lehetőségek

A kialakított struktúra és hálózati topológia alkalmas más tűzfalak tesztelésére is. Ezenkívül a Zorp topológiájának vizsgálata további érdeklődésre tarthat számot, lehet- a ZCV hatékonyságát növelni más komponens elosztással, a tartalomszűrést végző modulok másik hosztra való helyezésével. Míg ennek meg van a lehetősége, kérdés megéri-e?

5. Rövidítések

CIFS: Common Internet File System

DMZ:Demilitarizált Zóna

IDS: Intrusion Detection System

IPP: Internet Printing Protocol

IPS: Intrusion Prevention System

NAT: Network Address Translation

PAT: Port Address Translation

OSI: Open System Interconnection

PAM: Pluggable Authentication modul

PKI: Public Key Infrastructure

SMB: Server Message Block

SMP: Symmeetric mulitprocessing

Tos: Type of service

TTL: Time To Live

VNC: Virtual Network Computing

VPN: Virtual Private Network

6. Irodalomjegyzék

1. Designing Network Security, Merike Kaeo, 2004

2. Benchmarking Terminology for Firewall Performance, D. Newman, 1999
   http://rfc2647.x42.com/

3. Benchmarking methodology for Firewall Performance, B. Hickman, D. Newman, S. Tadjudin,T. Martin, April 2003
   http://community.roxen.com/developers/idocs/rfc/rfc3511.html

4. Firewall Performance Analysis Report, Chris Kostick, Matt Mancuso, 1995
   http://www.ussrback.com/docs/xperform.pdf.html

5. Evaluating Application-aware Firewall Performance
   http://advanced.comms.agilent.com/networktester/docs/whitepapers/pdfs/5989-1672EN.pdf

6. Guidelines on Firewalls and Firewall Policy NIST Special Publication 800-41
   http://csrc.nist.gov/publications/nistpubs/800-41/sp800-41.pdf

7. Firewall technologies http://www.boran.com/security/it12-firewall.html

8. Zorp administrator's guide http://www.balabit.hu/common-dl/zorp-admin-guide-3.1_v3.1.0.pdf

9. Zorp refence guide http://www.balabit.hu/common-dl/zorp-reference-guide-3.1_v3.1.0.pdf

10. Load and Performance Test Tools
    http://www.softwareqatest.com/qatweb1.html#LOAD

11. Load testing tools http://en.wikipedia.org/wiki/Load_and_performance_test_tools

12. Firewall Testing http://www.ixiacom.com/library/test_plans/display?skey=firewall

13. Firewall FAQ http://www.firewall-software.com/firewall_faqs/types_of_firewall.html

14. A Brief Taxonomy of Firewalls – Great Walls of Fire
http://www.giac.org/certified_professionals/practicals/gsec/0767.php

15. How to Implement a Content Filtering System
http://www.sans.org/rr/whitepapers/acceptable/1328.php

16. Is your firewall a bootleneck?
http://www.networkworld.com/newsletters/sec/0906sec1.html

17. Insight firewall performance testing
http://advanced.comms.agilent.com/n2x/docs/insight/2004-09/index.htm

18. Performance test uncovered
http://www.perftestplus.com/resources/perf_uncovered_ppt.pdf

19. System performance monitoring
http://www.comptechdoc.org/os/windows/win2k/win2kperformance.html

20. OpenSTA http://www.opensta.org

21. NetPerf http://www.netperf.org/netperf/NetperfPage.html

22. Siege hivatalos honlap http://www.joedog.org/JoeDog/Siege