Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar

Nagy Balázs

Adatközpontok hálózati forgalmának on-the-fly analízise SDN és IDS rendszerek támogatásához

Konzulensek

Dr. Varga Pál

Tóthfalusi Tamás

2016

TARTALOMJEGYZÉK

Kivonat

Az adatközpontok biztonsági követelményeinek teljesítése egyre nehezebb napjainkban. A hackerek egyre újabb és okosabb módokat találnak adatközpontok támadására, ezzel a jelenlegi védelmi megoldások határait feszegetik. A mindenki számára elérhető, és sok kiemelt ügyfelet is kiszolgáló adatközpontoknak a hálózati határán belül is meg kell küzdenie a saját különleges kihívásaikkal. Ilyen speciális problémák a belülről érkező támadások, valamint a rendszer kívülről történő, pontos feltérképezése.

A belülről érkező támadások különleges fenyegetést jelentenek, mivel kikerülik az adatközpontok elsődleges védelmi vonalát, a root switchek tűzfalait. Az adatközpontok topológiájának (fat and short) köszönhetően komoly gondot jelent a belső támadások észrevétele és megakadályozása. Ezen támadások nemcsak üzleti, hanem fizikai veszélyt is jelentenek a rendszerre.

Az integrált biztonsági rendszerekbe speciális hardver-támogatás (például FPGA, Field-Programmable Gate Array) beiktatása olyan lehetőségeket ad, amire sima szoftver-alapú rendszerekben nincs lehetőség. A FPGA olyan nagysebességű párhuzamos adatfeldolgozást nyújt, ami gyorsabb, okosabb reagálást tesz lehetővé a rendszert ért támadásokkal szemben. A tisztán szoftver-alapú rendszerek reakcióidejének töredéke alatt képes észlelni a támadásokat, és elküldeni a switch-eknek a támadó adatait: ezzel segíti a rendszer védelmét. Ez különösen fontos úgy nevezett hit-and-run támadások esetén.

A dolgozat célja annak bemutatása, hogy egy ilyen FPGA-t is tartalmazó biztonsági rendszerben hogyan lehet különböző szintű és kifinomultságú (D)DoS (Distributed Denial of Service) támadásokat észlelni és leállítani. Ezen dolgozatnak nem célja, hogy az összes típusú támadásra mérnöki megoldást adjon. A dolgozat egy olyan, könnyen bővíthető, flexibilis keretrendszert mutat be, ami a leggyakoribb Denial of Service támadások hardver-támogatott észrevételére és továbbterjedésének megakadályozására szolgál. Emellett útmutatást ad arra is, hogy hogyan lehet másfajta támadásokhoz megfelelő modulokat tervezni. A koncepció működésének validálására egy kiterjedt magyarországi hálózat (NIIFI, Nemzeti Információs Infrastruktúra) által nyújtott adatközponti szolgáltatást ért támadások valós forgalmi mintáit használtuk fel. Mivel a forgalom-analízishez használt hardveres gyorsítással a hálózat állapotáról is finom felbontású információhoz jutunk, az architektúra az SDN (Software Defined Networking) vezérlők döntéshozásának támogatására is alkalmas.

Abstract

On-the-fly Traffic Analysis of Data Center Networks to Support SDN and IDS Systems' Decision Making

Making data centers secure and reliable becomes harder and harder these days. New ways are continuously created to exploit the data center network's weaknesses, making yesterdays defensive solutions obsolete. Data centers (DCNs) designed for public and VIP use come with their own set of vulnerabilities. These include the attacks that originate from the inside the data centers, and the attacks aiming for completely discover of DCN's architecture, including their hardware setup.

Attacks originating from inside the DCN pose a special threat, because these kind of attacks avoid the data centers' primary defense line: the firewall of the root switches. The DCNs unique topology (fat and short) make it really hard to detect and stop attacks on the network level. These attacks pose not only financial but physical threat to the system.

Introducing a hardware-acceleration support such as FPGA (Field-Programmable Gate Array) into the integrated defense solutions, creates options that weren't possible in pure software based systems. The FPGA's ability to parallel process data at extremely high speeds can make the defense system smarter, and quicker. A solution using such FPGA-based hardware acceleration can react to attacks in a fraction of the second, making it superior to pure software based systems. This kind of speed and agility is becoming more and more relevant, as hackers create new schemes, such as hit-and-run attacks.

This paper's purpose is to showcase a hardware accelerated, monitoring-based information provider system that supports intrusion detection systems (IDS) as well as the software defined networking (SDN) paradigm. As a case study, the paper proves the concept's ability to detect and shutdown (D)DoS (distributed denial of service) attacks of different layer and complexity. This paper does not provide engineering solutions for all types of attacks. Still, it showcases a flexible, extensible system and solutions for the most common attacks, and a guide to design modules for other type of attacks. The validation process includes the usage of traffic patterns recorded during an attack against (NIIFI, (Hungarian) National Information Infrastructure) and the usage for some of the most popular attack tools designed bringing down servers. The hardware's data processing ability provides us a high resolution image of the switching system, which can be used to support SDN (Software Defined Networking) controllers' decision making.

Köszönetnyilvánítás

A dolgozatom nem jöhetett volna az IBM Research - Zürich munkatársai kutatása és munkája nélkül, szeretnék nekik, különösen Dr. Mitch Gusat -nak köszönetet mondani a lehetőségért, hogy velük dolgozhattam.

Acknowledgements

The author would like to thank the employees of IBM Research - Zürich, especially to Dr. Mitch Gusat for the opportunity to participate in the project. The author appreciates the provided monitoring scheme and the monitoring results that this paper is based on.

1. Bevezetés

Az elmúlt években lényegesen átalakult a cloud alapú rendszerek és szolgáltatások profilja. Az új fejlesztések nem csak a felhasználóknak teremtettek új lehetőségeket. Az internetes hálózatok fejlődése mellett sebességben és technológiai újításokban elmaradt a monitorozási és védelmi technológiák fejlődése. Ha most nem fejlesztjük tovább a biztonsági eszközeinket, nagyon komoly árat kell fizetnünk a jövőben.

Ennek a dolgozatnak a célja bemutatni egy csúcstechnológiás, kísérleti biztonsági rendszert, amelyet az adatközpontok biztonsági kihívásai hívtak életre, és amely az SDN hálózati architektúra támogatására is használható. Egy könnyen installálható plug and play rendszert, ami átalakíthatja az adatközpontok védelmét. A dolgozat egy egyszerű példán keresztül mutatja be a rendszer hatás-mechanizmusát.

A rendszer új ötletek, csúcstechnológiás hardware, és software fúziójából jött létre. Az FPGA alapú hardwarenek köszönhető a rendszer csúcstechnologiás sebessége és teljesítménye. A zárt vissza csatolási körön alapuló nem invaziv elrendezésnek köszönhetően, jelenlegi rendszerektől különbözően, nem visz késleltetést a kommunikációba.

2. Az adatközpontok felépítésének jellegzetességei

A számítástudomány fejlődésével és a szolgáltatás-felügyeleti módszerek optimalizálásától hajtva egyre több alkalmazás kerül adatközpontokba. Ez a trend a jövőben csak gyorsulni fog, többek között az IoT (Internet of Things) eszközök elterjedése, és az ezzel kapcsolatos szolgáltatások miatt. Emellett ma már majdnem mindegyik okostelefonos alkalmazás használ valamilyen cloud szolgáltatást.

Az adatközpontok fejlődése nemcsak pozitív kicsengésű figyelmet hozott ennek a témának. Ez a központosított architektúra sosem látott lehetőségeket adott a hackerek kezébe. Egyetlenegy szerverpark megtámadásával millióknak tudnak kárt okozni, gondoljunk csak a dolgozat bemutatása előtti napokban (2016.10.21.) megvalósult nagy, DNSközpont elleni támadásra[1], egy támadásra ami elérhetetlenné tette az USA több legnagyobb tartalom szolgáltató oldalát. Az adatközpontok elleni legkézenfekvőbb támadás az úgynevezett (D)DoS (Distributed Denial of Service). Az ilyen támadásoknak nem célja a bonyolult biztonsági rendszerek feltörése, hanem csak a szerver ideiglenes túlterhelése.

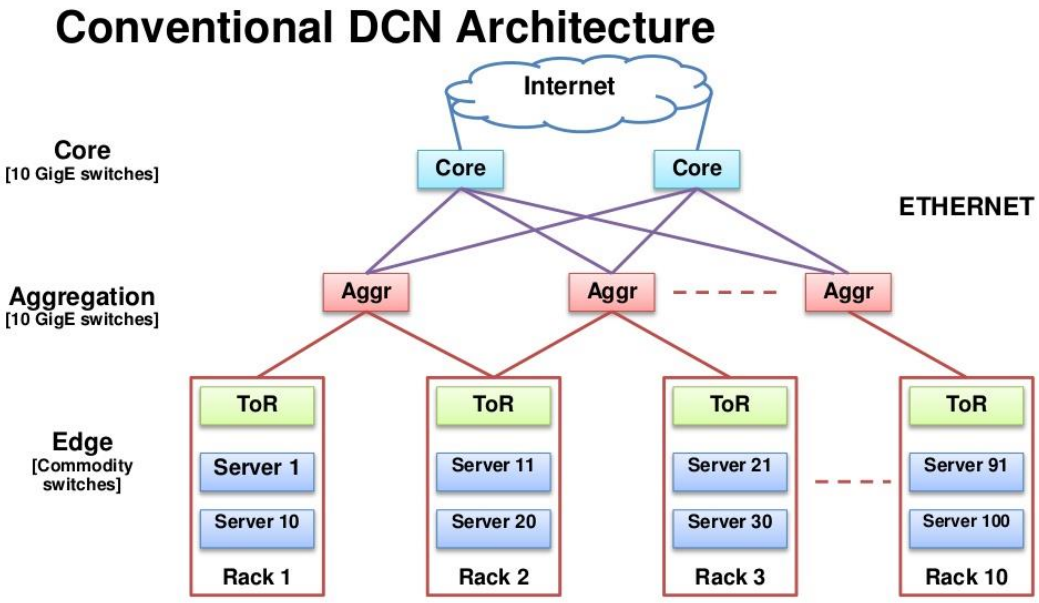
2.1. Adatközpontok architektúrája

Az adatközponti eszközök manapság fa-struktúrában vannak felfűzve [2]. Adatközpontok esetén ezt a topológiát „Fat-and-short”-nak nevezzük, mivel kevés szintje van (short) és nagy sáv szélességűek a belső kapcsolatok (fat). A legalsó szinten vannak a szerverek, ahhoz kapcsolódnak a ToR (Top of the Rack) switchek, afölött vannak az aggregátor avagy összefogó switchek, a legfelső szinten pedig a root switchek (core router). Ilyen elrendezésekre mutat példát az **Hiba! A hivatkozási forrás nem található.**.

A root switchekbe épített biztonsági rendszerek dolga az, hogy a kívülről érkező támadásokat megakadályozza. Ezeket a switcheket több évtizedes tapasztalattal tervezték csúcstechnológiás hardver és szoftver felhasználásával. Ebben a kategóriában nagyon nehéz olyat mutatni, ami a már kommerciális forgalomban lévő eszközökkel fel tudja venni a versenyt.

Figyelemre méltó azonban, hogy nem minden forgalom halad át ezeken a root switcheken. Az 1. ábra-n láthatjuk, hogy a belülről jövő támadás sok esetben csak egy, vagy két switchen halad át. A támadó akár publikus központokban is tud magának bérelni egy virtuális gépet, amit felhasználhat egy másik bérlő támadására.

Ezen dolgozat egyik kiemelt célja, hogy az adatközpontokat fenyegető **belső** támadások elleni védekezésre megoldást adjon.



1. ábra Általános adatközpont architektúra [21]

2.1.1. Az adatközpontok elemeiToR switchek

Ezek kommerciális, nagy darabszámban gyártott, költséghatékony switchek, amik egy rackben található szerverek összefogására használnak [14]. Általában kicsi buffer mérettel rendelkeznek.

2.1.2. Core router

Az adatközpont hierarchiájának a csúcsán álló switchek. Nagyon nagy teljesítményűek, drágák előfordul, hogy a hardware vagy software a vevő igényei szerint specializáltak [14]. Általában a gyártók csúcs kategóriás termékei kerülnek ebbe a kategóriába. Kis hálózattokban csak egyet használnak, nagyobb, biztonságosabb hálózattokban többet [15].

2.1.3. Védelmi rendszerek

A védelmi rendszerek általában a core routerek tűzfalából és a nagyobb exploitok kezeléséből (mitigálásából) tevődik össze. A core routerek tűzfalai nagy kapacitású, csúcstechnológiájú védelmi eszközök. A core routerek tűzfala nagy biztonsággal megállítja kívülről érkező támadásokat. Az exploitok mitigálásán azt kell érteni, hogy az adatközponti hálózatok (Data Center Networks, DCNs) tervezésékor számoltak különböző gyengeségekkel, amit a hálózat softwares és hardwares elemei és a szabványok okoznak. Ezekben a hálózatokban is a szabványok alapján dolgoznak, viszont a megvalósításnál igyekeznek csökkenteni a protokollok gyengeségeit, sebezhetőségét. Jellemző példa erre, hogy a DCN topológia legalján kiszűrik a spoof-olt csomagokat (amikor a feladó nem a saját azonosítójával látja el a csomagot), vagy az ICMP (Internet Control Message Protocol) üzenetek küldési sebességét korlátozzák.

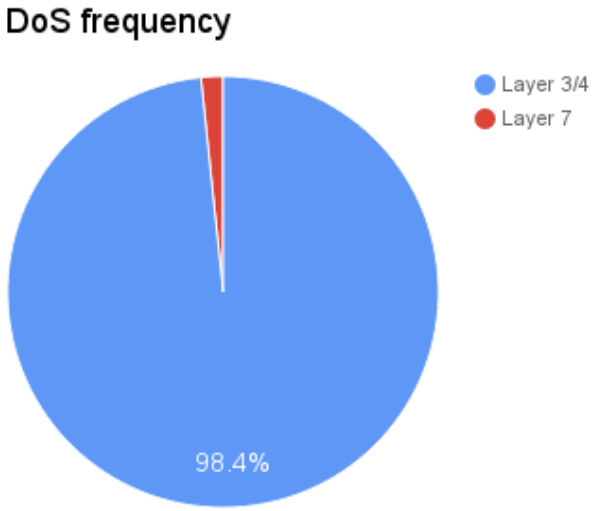
2.1.4. Szerverek

Ezek az eszközök futtatják a rendszer funkcionális applikációit. Nagy teljesítményű, több processzoros architektúrákon alapulnak. A szerverek hardware-e nem csak teljesítményben, de minőségben is kiemelkedő. Egy fizikai szerver 1-200 virtuális gépet futtat párhuzamosan. Tipikusan magukat a virtuális gépeket kölcsönözi ki az üzemeltető, publikus hálózati szolgáltatás esetén, azaz a szerverek processzorait és memóriáját – valamint a hozzájuk kötődő biztonsági és hálózati szolgáltatásokat teszi pénzzé. Önmagában a hálózati erőforrásokat nem, vagy csak nagyon ritkán árulják.

3. A (Distributed) Denial of Service típusú támadások

A DDoS támadás-típus az internet hajnala óta létezik; modern formáját a kétezres évek elején nyerte el. Az első nagy méretű DDoS támadás 1999-ben történt a Minnesotai egyetem ellen: az egyetem szerverei napokra leálltak. Jelenlegi „népszerűségét” igazán 2010 után nyerte el, amikor a hírhedt hacker csapat, az Anonymous elkezdte használni ezt a metódust [13]. A DDoS-t mint módszert használják politikai aktivizmusra, zsarolásra és bosszúra is. Az elmúlt években ugrásszerű növekedésen estek át a DDoS támadástípusok, 2016 és 2015 között fantasztikus 139 százalékos növekedés volt[3].

Nagyon sok különböző DDoS támadás-típus létezik, ezeknek egy közös ismertetője van: az, hogy az áldozat valamely erőforrását teljesen fel akarják használni. Mindezzel azt igyekeznek elérni, hogy az áldozat ne tudja kiszolgálni jószándékú klienseit. A DDoS támadásokat a szakirodalom két részre osztja: (i) Layer 3,4 és (ii) Layer 7.

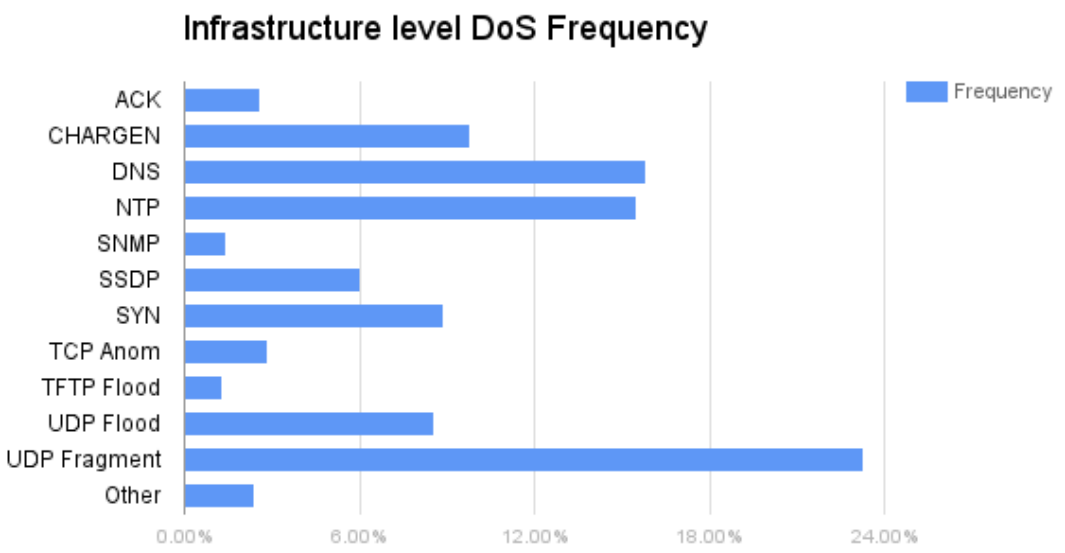


2. ábra (D)DoS támadások eloszlása 2016 Q2 [3]

3.1. Layer-3/4 támadások

A Layer-3/4 támadások a legegyszerűbbek és a leggyakoribbak, Layer 7 támadásokhoz képest nagyobb mennyiségű adat átvitelére hagyatkoznak. Van olyan támadás, ami csak az áldozat hálózatát akarja túltelíteni, van olyan, ami a processzor vagy memória kapacitásainak fogyasztására játszik. Leggyakoribb fajtái: UDP flood[4], SYN flood[5], UDP fragmentation[6], TCP fragmentation[6]. Ezek a támadások csomagszinten primitívek de nagyon hatékonyak tudnak lenni ha jól használják.

3.1.1. Layer-3/4 támadások eloszlása



3. ábra Layer-3/4 támadások eloszlása 2016 Q2[3]

A

3. ábra Layer-3/4 támadások eloszlása 2016 Q2[3]

az Akamai 2016 Q2-s kutatása[3] alapján készült. Ez a „nagy” internet támadás-fajtáit foglalja össze, nem feltétlenül releváns az adatközponti hálózatok (Data Center Networks, DCN) belső támadásaira. Mindenesetre megközelítőleg jó képet ad arról, hogy milyen támadások terjedtek el.

3.1.2. SYN flood

Ez az egyik legrégebbi támadási forma, 1996-ban vált ismerté a nagyközönség számára. A TCP protokoll gyengeségét használja ki. A TCP egy biztonságos kapcsolat orientált protokoll, cserébe sok erőforrást (elsődlegesen memória, másodlagosan CPU) használ. A SYN flood arra épít, hogy a támadó kevés erőforrás felhasználásával újabb és újabb kapcsolatok nyitására készteti az áldozatot azért, hogy ezzel erőforrásokat vonjon el a valós kapcsolatoktól[9]. Ez a támadás az aszimmetrikus hadviselés elvét használja: egy egységnyi támadó oldali erőforrás több egységnyi áldozat oldali erőforrást használ. Sajnos ezen tulajdonságok miatt nincs úgynevezett „ezüstgolyó” a SYN flood ellen. Általában szerver oldali megelőző tervezéssel igyekeznek kezelni a támadás hatásait. Léteznek hálózati technológiákon alapuló védelmek is, itt tűzfalakat használnak.

3.1.3. UDP flood

Az UDP flood a primitiv volumetrikus támadások állatorvosi lova. A támadó véletlenszerűen kiválasztott portokra küld UDP csomagokat. Az áldozat operációs rendszere megnézi, hogy van-e nála olyan applikáció, ami azon a porton figyel – és ha nincs (általában nincs), akkor egy ICMP üzenetet küld, amiben jelzi: nincs célalkalmazás az adott porton. Ennél a támadásnál a támadó nagyon rossz hatásfokkal tudja felhasználni az erőforrásait, viszont minimális szaktudással és előkészülettel is hatásos támadást lehet lefolytatni egy felkészületlen rendszer ellen. A rossz hatásfok miatt erre a célra botneteket szoktak használni.

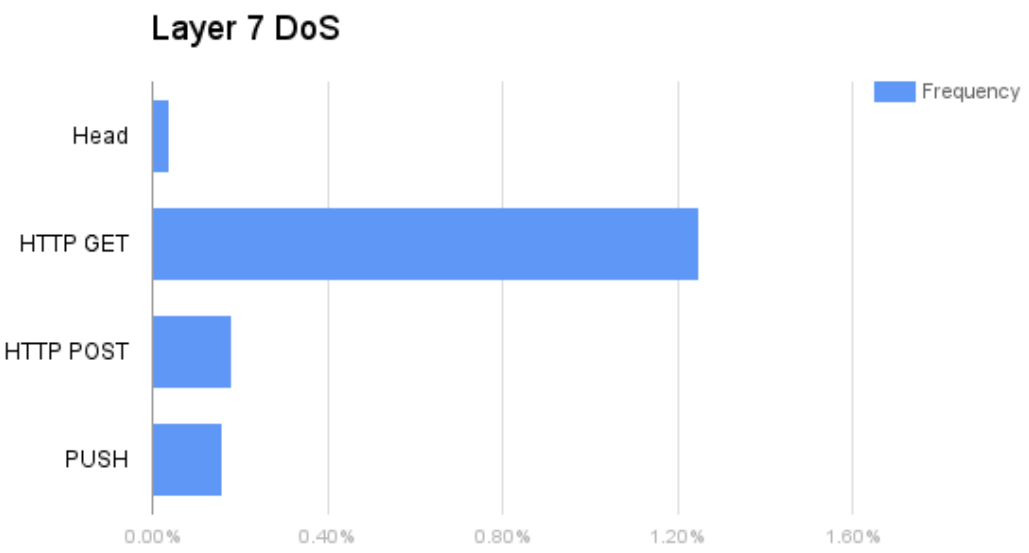
3.1.4. UDP fragmentation

A leggyakoribb Layer 3,4 támadás [3],[8]. Azt a tényt használja ki, hogy a nagy datagramokat az adatkapcsolati réteg korlátai miatt fel kell darabolni. A feldarabolt datagramokat a fogadó host rakja össze. Ha tehát a támadó egy hamis datagram-darabot küld, azt a host elraktározza és megpróbálja összeállítani a többi datagram-darabbal – ami persze nem sikerül, mivel a támadó nem küldi el a datagram többi részét. Ez a támadás az áldozat CPU-ját és memóriáját fogyasztja leginkább; a népszerűséget egyszerűségének és jó hatásfokának köszönheti. A jelenlegi védelmek főleg host-oldalon próbáljak kezelni a támadás hatásait.

3.2. Layer 7 támadások

A layer 7 támadások már sokkal kifinomultabb módszereket használnak a céljaik elérésének érdekében. Ezek a támadások az applikációk gyengeségeit használják ki, az előbbiekhez képest sokkal kevesebb támadói hardwaret igényelnek, viszont sokkal nehezebb őket implementálni, mivel itt a gépi erőforrások hatásfokának növelése a cél. Egy ilyen támadás megtervezéséhez pontosan ismerni kell a szerver oldalon lévő szoftvereket, viszont akár egy okostelefonról is le lehet vezényelni őket. Ezek a támadások jóval kevesebb csomagot küldenek, mint a Layer-3/4 jellegűek, ezért sokkal nehezebb őket a hálózati architektúrában megfogni, kezelni. A Layer 7 támadások még nem terjedtek el tömegesen. Ennek oka részben az, hogy meglehetősen könnyű nagy sávszélességet szerezni Layer-3/4 támadásokhoz, és azok használatával is megfelelő mértékű kár okozható a támadók részéről [10].

3.2.1. Layer 7 támadások gyakorisága



4. ábra Layer 7 támadások eloszlása 2016 Q2 [3]

3.2.2. HTTP get, post

A HTTP alkalmazások általában két paranccsal kommunikálnak: a GET-et általában statikus tartalom elérésére, a POST-ot dinamikusan generált tartalmak elérésére használják. A http támadásoknak két elterjedt formája van.

1. Teljesen legitim lekérdezéseket hajt végre a támadó sok hostról, általában a leginkább erőforrás-igényesebbeket. Ehhez botneteket használnak, hogy volumetrikus szűrésen ne bukjanak le.
2. Elküld egy legitim GET/POST headert a támadó. Az üzenet body részét nagyon lassan küldi el, ezzel a szervert ráveszi, hogy nyitva tartsa a kapcsolatot. Ez sokkal hatásosabb, viszont lényegesen könnyebb észrevenni.

3.3. (D)DoS védelem

A (D)DoS védelemnek több formája terjedt el, különböző szintű eredményességgel. A DDoS támadások sokfélesége miatt nem létezik univerzális védelmi megoldás. A legegyszerűbb „házi foltozástól” a szofisztikált ASIC-alapú megoldásokig rengeteg védelmi rendszer készült az elmúlt évtizedekben. A legelterjedtebb védelmi módszerek:

1. Szignatura analízis: a csomagokban ismert mintákat keres, vagy a csomagokat egymáshoz hasonlítja [18].
2. Forgalom analízis: a forgalomban anomáliákat keres ami támadásra utalhatnak.
3. Reputáció alapú rendszer: egy központi adatbázisban tárolja az ismert támadók IP címeit, botnetek ellen hatásos [12].

4. Host letapogatás: A kapcsolatainak különböző teszteket küld amivel kideríti, hogy tényleg támadó-e, pl: captcha.

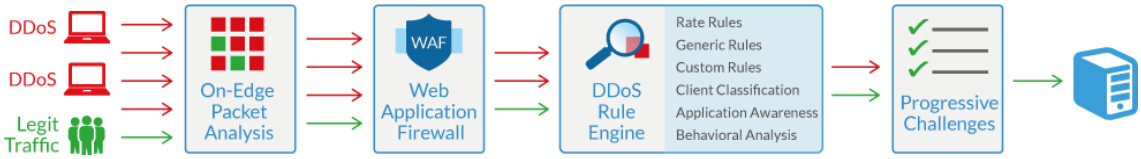
A csúcstechnologiás rendszerek ezek közül többet is használnak, feltehetően nem publikált saját megoldásokkal kiegészítve.

3.3.1. Egyedi, host-oldali megoldások

Ezeknek a megoldásoknak nem a támadások észrevétele vagy megakadályozása célja, hanem a támadó hatásfokának csökkentése. Az alkalmazás-fejlesztő dolga az, hogy az applikációjában ne hagyjon olyan lehetőségeket, amiket egyszerűen ki lehet használni. Jellemző példaként, ha a webszerver 200 kérést tud egyszerre feldolgozni, és 10 perc a timeout ezeken a kapcsolatokon, egy GET támadás 20 csomag/perccel letudja kapcsolni a rendszert. Ha levenné a rendszer üzemeltetője 6 másodpercre a timeoutot (ez még mindig nagyon sok), a támadónak 100-szor több erőforrást kell használnia. Körültekintő programozással el lehet venni a legtöbb támadó kedvét, mivel a legtöbb támadó nem szakértő, hanem egy laikus, az internetről letöltött sok éves támadó eszközzel. Sajnos még a körültekintő „házi” programozás is hatástalan a komolyabb támadások ellen – ott összetettebb védelemre van szükség.

3.3.2. Imperva Incapsula

Az Imperva üzemelteti a világ legnagyobb, legsikeresebb DoS védelmi szolgáltatását [16], nagyon összetett védelmi rendszerrel dolgoznak. Jelenleg ők képviselik a behatolásészlelés és -mitigálás csúcsát.



5. ábra Incapsula védelmi rendszere [17]

Az Incapsula többszintű védelemmel éri el azt a hatásfokot, ami piacvezetővé tette. Ez a rendszer jelenleg a DDoS védelem csúcstechnológiája, legalábbis kereskedelmi szinten. Egyrészt proxyként viselkedik a kliens számára, ezzel elrejti a felhasználói IP címét a nagy közönség elől. Az Incapsula a hagyományos forgalom alapú analízis mellett reputáció alapú adatbázist is használ a DDoS botok kiszűrésére [17]. Ez a rendszer nagyon komoly késleltetést visz a kommunikációba, és erősen invazív. Ahhoz, hogy ez a rendszer jól működjön, minden egyes csomagnak át kell menni az Incapsulán. A rendszer üzemeltetői ezen felül azt tanácsolják a klienseiknek, hogy minden, az Incapsulán kívülről érkező csomag-továbbítást tiltsanak le [17], ezzel a memória és processzor ellen irányuló támadásokat ellehetetlenítik. Ha a kliens IP címét megszerzik a támadók, elárasztható a klienst, vagy a hálózati eszközeik kapacitását is kimeríthetik. Az Imperva védelmi rendszere ezen felül több nagy erejű szerver parkot üzemeltet, hatalmas sávszéleséggel. Erre a volumetrikus támadások elnyelése miatt van szükség. Volumetrikus támadások esetén az ökölszabály, hogy ha a támadónak nagyobb a támadása mint a védelmi rendszer hálózati kapacitása, akkor valamilyen szinten sikeres lesz a támadás.

3.4. (D)DoS adatközpontok belsejében

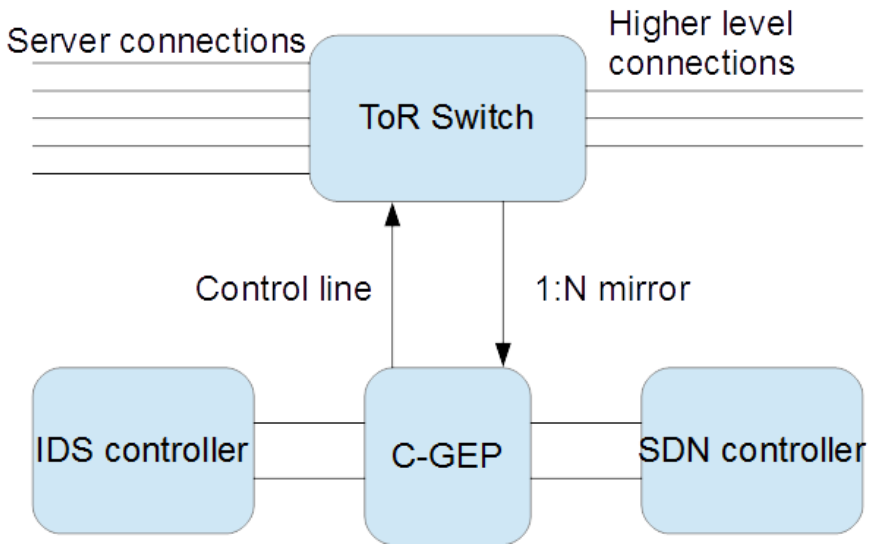
A (D)DoS támadások adatközpontok belsejében lényegesen különböznek a központokon kívülről induló támadásoktól. A feltételek különbözősége ennek az oka: sokkal nehezebb a támadáshoz hostot találni ezekben a központokban, mint az interneten; sokkal nehezebb központokban eltüntetni a támadó kilétére utaló nyomokat, a központokban a hálózati kapacitás nagyon nagy[7]. Viszont adatközpontok belülről érkező támadások elleni védelme még csak a „házi megoldások" fázisban van, és az adatközpont komponenseit részletesen fel tudják a támadok térképezni. A Microsoft és az Amazon csak nemrég kezdte el korlátozni az egy virtuális géphez tartozó sávszélességet [7].

4. Visszacsatolt védelmi rendszer

A dolgozat nagy újdonsága, hogy az -placeholder- aktív visszacsatolási kör felhasználásával készült. Ez egy egyszerűen telepíthető, üzemeltethető rendszer. Az alapötlete az, hogy egy switchen, vagy switch rendszeren átfolyó adatott átadjuk egy feldolgozó és döntéshozó rendszernek. A FPGA alapú platformnak köszönhetően az adatfeldolgozás sebessége micro secundum nagyságrendben lesz, ezzel valós idejű beavatkozást tesz lehetővé. Ez a megoldás nem invaziv, ha a hálózat megfelelő switcheket használ minimális beavatkozással lehet telepíteni a rendszerbe. Switchenként egy linket szabaddá kell tenni a mirrornak. Hasonló funkciónalitású rendszerek instalálása, csúcskategoriás switchek behelyezésével, vagy a hálozat teljes újra tervezésével (nem biztos, hogy lehetséges) járna. Ez hatalmas újdonság, mivel lényegesen lecsökkenti a telepítés költségeit. Az adatközpontos példán értelmezve, régi megoldásokkal minden ToR switchet csúcskategoriásra kellene cserélni.

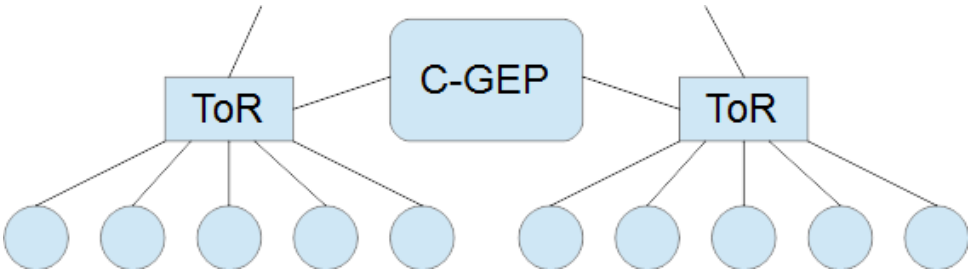
4.1. A rendszer felépítése

A switchek mirror funkciót használva lemásolják a switch forgalmát, ezt a forgalmat átadjuk a C-GEP-nek. Természetesen a mirroron lemásolt forgalmat megcsonkítva adjuk át a feldolgozónak, a csomagok payload részét levágjuk. A csonkolásra azért van szükség, hogy megnöveljük a mirroron átvihető csomag mennyiséget. A C-GEP feldolgoza ezt a forgalmat és kiszolgálja az IDS szoftwaret, és az SDN controllert.



6. ábra A védelmi rendszer blokkvázlata

Ezt a rendszer minden ToR switchével megcsináljuk. A kommunikáció minden esetben etherneten megy, viszont hatalmas különbség van a sávszélességben. A mirror 100G-n küldi az adatokat, a switch controlja nem használ többet 1kb/s.



7. ábra A rendszer több switchhel

4.2. A rendszer időszerűségé

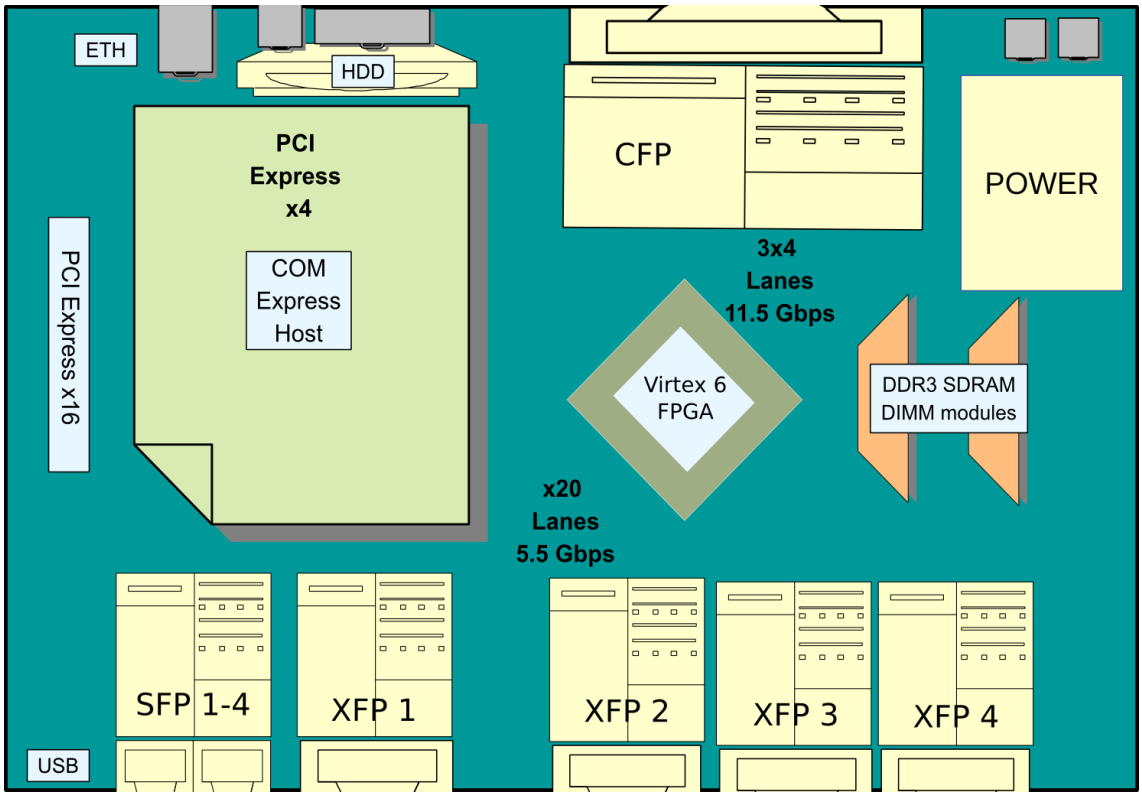
A múltban egy ilyen rendszer elképzelhetetlen lett volna. A bolti switchek mirror funkciója nem volt képes egy ilyen feladat ellátására, mirrorok ezekben a switchekben nagyon komoly kompromisszumokkal készültek. Ezen felül a switchek csomagcsonkolás és időbélyegző funkcióival is komoly gondok voltak. A megfizethető kategóriában nem létezett olyan switch ami ezt a három funkciót egyszerre kielégítő szinten tudja kezelni. A rendszer egy még kereskedelmi forgalomba nem hozott switchet használ. Az adatfeldolgozás kapacitásának fejlődésé is csak elmúlt években tette lehetővé ilyen mennyiségű adat feldolgozását. Most jött el az idő, hogy a kör bezáruljon. A switchek eljutottak arra a szintre, hogy megfelelő minöségben tudják adat szolgáltatni. A FPGA alapú adatfeldolgozás arra, hogy valósidőben reagáljon.

4.3. A rendszer funkcionalitása

A rendszer nem csak telepítésében egyszerű, de funkcionalitását tekintve is sok oldalú. A rendszer alkalmas SDN döntés hozás támogatására, a nagy felbontású, de könnyen feldolgozható információival. A rendszer alkalmas minden fajta forgalom anomália detektálására, mint például DDoS támadások. A rendszer verifikálására én egy DDoS egyszerű DDoS támadást választottam.

4.4. C-GEP

A C-GEP az AITIA IA. által gyártott FPGA alapú 100 gigabit/s ethernet kapacitású csomag feldolgozó platform 8. ábra. A C-GEP egy Virtex 6-os Xilinx FPGA-t használ, a csomagfeldolgozáshoz. A nagy erejű újra konfigurálható hardwarenek köszönhetően képesek is vagyunk feldolgozni a forgalmat ami érkezik a 100G-s linken [11]. A FPGA-ra implementált cél hardware teljesítményével csak nagyon drága és komplex felhő alapú rendszerek tudnak verszenyezni. A FPGA 315 Mhz-en üzemmel, ez a magas frekvencia lényegesen megnehezíti és lelassítja a tervezést. A FPGA erőforrásai sokkal korlátoltabbak, mint egy PC alapú rendszernek, valamint sokkal nehezebben bővíthetők. Ezért célszerű más elemeket is bevonni a rendszerbe.



8. ábra C-GEP blokk diagram [22]

4.5. C-GEP Switch communication protocol

A C-GEP-nek és a switchnek célszerű kommunikálnia, ha alapvető passzív megfigyelésnél komolyabb feladatokat akarunk ellátni. A tervezésénél fontos volt hogy egyszerű és portolható legyen. Ipv4/UDP protokolokra esett a választás, tehát UDP csomag fogja tartalmazni az irányító üzenetet. Ezek az üzenetek elsősorban a switch konfigurálására valók. A hálózat adatfolyamába a védelmi rendszer csak közvetve tud beleszólni, mert nem folyik át rajta forgalom, ezért szükséges a switch vezérlése.

4.6. Az IDS software

Az IDS software komponense nincs meghatározva. Legfontosabb követelmény vele szemben, hogy nyílt forrás kódú legyen és könnyen módosítható.

4.7. A switch

A switchet nem nevezem meg, de funkcióit és kiválasztásának logikáját leírhatom. Nagyon fontos szempont volt, hogy erős mirror funkcióval legyen ellátva. A rendszer képtelen lenne müködni egy jó mirror nélkül. Megkell tudnia csonkolni (truncating) mirroron kiadott csomagokat, és timestamppel ellátni. A switch elterjedtségé is nagyon fontos szempont volt.

4.7.1. A mirror

A mirror eredeti célja az volt, hogy a mienkhez hasonló rendszereket lásson el adattal. Ez nem egy új technologia, viszont bolti switcheknél nem volt prioritás. --------

4.7.2. A csomagcsonkolás

Nagyon fontos követelmény a csomag csonkolása. Az analízisekhez amiket végzünk nincs szükség a csomag testére, nem érdekel hogy miről kommunikálnak (sok esetben titkos is) csak az, hogy hogyan. A switcheken a sávszéleség értékes erőforrás, ezért minél kevesebbet fogyasztunk annál „olcsóbb" a rendszer. Ideális lenne ha képes lenne a csomag fejléc hosszát felismerni a switch és a fölött vágni, reálisan az is elég ha egy bizonyos konfigurálható bit hossz felett vág.

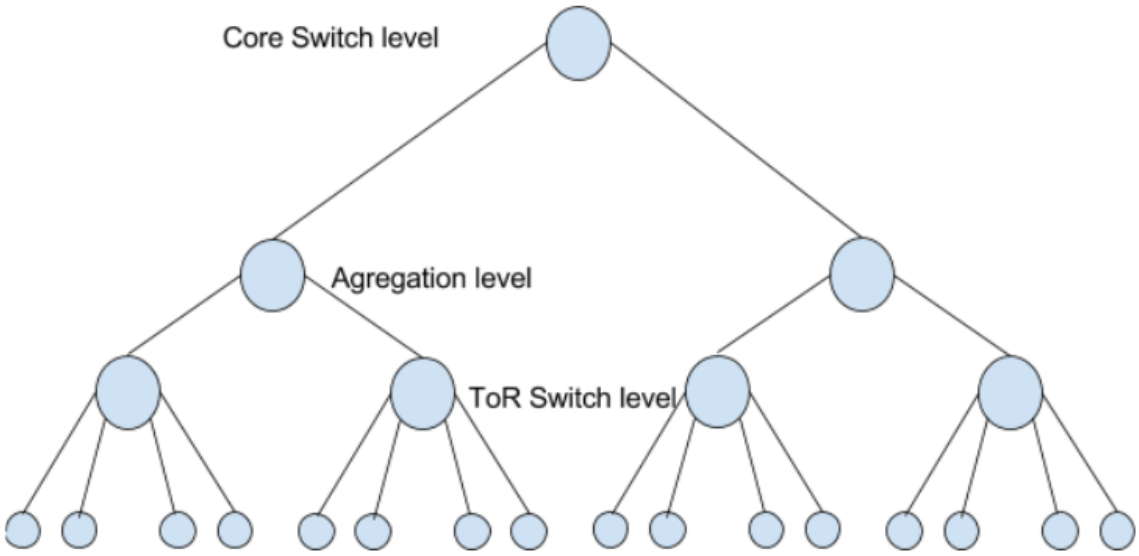
4.7.3. Az időbélyegzés

A timestamp (időbélyeg) talán a legkevésbé magától értetődő. Ha rendszert egy nagyobb hálózatra akarjuk instalálni, nem csak egy ponton kell mintavételeznünk a hálózatból, nem csak egy C-GEP-pel. Ahhoz, hogy koherens képet kapjunk a hálozatról a feldolgozandó csomagokat nem csak térben (MAC) de időben is pontosan elkell tudjuk helyezni. Szükséges tehát eltárolni, hogy adott csomag mikor „volt" a switchen ahol mintavételeztük. Ideális lenne ha egy ha a switchek egy szinkronizált nagy pontosságú óra szerint a beérkezés pillanatában a csonkolt csomag végéhez hozzáillesztené a timestampet. Suboptimális megoldás lehet, ha a feldolgozó C-GEP-ek timestampelnék a csomagokat. Ezzel az a probléma, hogy a switch mirrorján nem feltétlenül sorrendhelyesen adja ki a csomagokat, maga a mirrorozás rakna egy kisebb bizonytalanságot az időbélyegzésbe.

4.8. A SDN controller

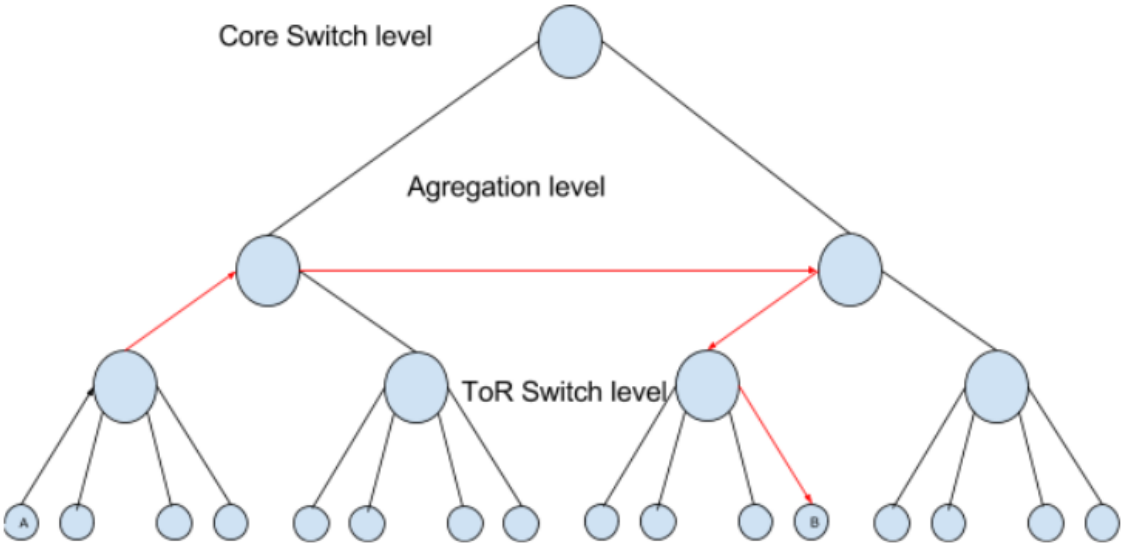
Jelenleg a rendszer nem működik együtt SDN rendszerrel. Ha tartalmaz SDN-t a hálozat, kiválóan alkalmas annak a döntés hozatalát segíteni. A switchek (különösen az olcsó ToR switchek) limitált segítséget tudnak nyújtani a SDN döntéshozásban, ezen ezközök elsődleges feladata a csomagok gyors és költséghatékony switchelése. Az FPGA alapú C-GEP platform ezzel szemben, gyorsabban jobb nagy felbontású információt tud küldeni. A SDN-nek a döntéshozatalához minél részletesebb információra van szüksége, ebben a C-GEP adat feldolgozó platform hatalmas segítséget nyújthat. A SDN a támadások effektív kiszűrésében hatalmas segítséget nyújthat. A SDN-nek köszönhetően a (D)DoS támadás csomagjait a támadóhoz sokkal közelebb tudjuk kiszűrni.

5. Adat mennyiség optimalizáció nagy hálózattok esetén



9. ábra Egyszerűsített példa az adatközpont fa architektúrájára

A védelmi rendszer egyik célja a belülről érkező támadások felismerése és megállítása. Ehhez szükséges minden központon belüli szerver-szerver csomag megfigyelése 9. ábra, ezt fizikailag a switchek mirror funkciójának segítségével valósítjuk meg. A switch minden csomagot amit megkap és elküld kirakhat a mirrorra is. Lenti példán láthatjuk, A-bol B-be tartó csomagot négyszer fogjuk feldolgozni 10. ábra. Ez hatalmas erőforrás pocsékolást eredményez. Ha elérjük, hogy egy csomagot csak egyszer dolgozzunk fel, sokkal olcsóbbá tehetjük a rendszer adat feldolgozási részét. Célszerű lenne egy jó matematikai modellt alkotni a problémára.



10. ábra A-ból B-be tartó csomag útja

Egy jó modell elkészítése, ami implementálható hardwarere nem könnyű. Célszerű több lehetőséget is megvizsgálni mielőtt választunk.

5.1. Adatközpontokra jelemző tulajdonságok

A matematikai model megalkotásánál muszáj a következő adatközpontokra jellemző tulajdonságokkal számolni:

Egy ToR switch-hez 1-60 szerver kapcsolodhat.

ToR switchek közt lehet direkt átkötés.

Egy virtuális gép egy switchel van összekötve.

Agregációs szint több fizikai szintből is állhat.

Agregációs szint bármely két switch-e össze lehet kötve.

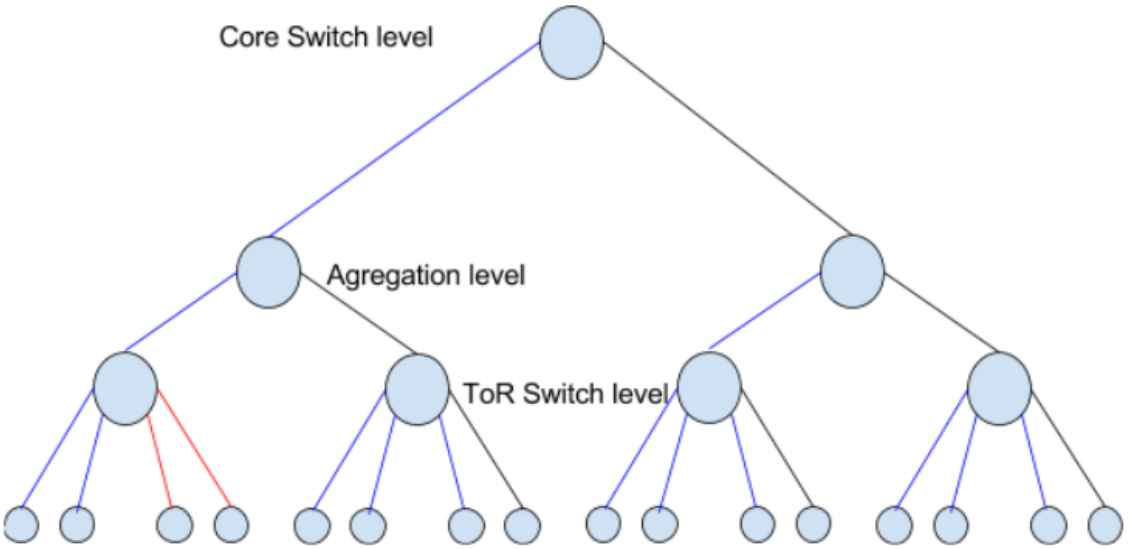
Szerverek direkt nem lehetnek összekötve.

A központ üzemeltetői bármikor változtathatnak a topológián, a funkcionalitás fenntartására figyelve.

5.2. Modellezés irányítatlan gráffal

Ha irányitatlan gráffal modellezzük a hálózatot a legjobb lehetőségünk van, rackenként az összes mínusz egy szerver-ToR él lefogása, és néhány felsőbb él lefogása.

Amennyiben rackenként kevesebb élt figyelünk könnyen belátható lesz olyan szerver-szerver utak amit nem fogunk le 11. ábra (kék a figyelt élek, piros a nem figyelt út).



11. ábra Irányítatlan modell

5.3. Modellezés irányított gráffal

Irányítatlan gráf a legkézenfekvőbb modell, kérdéses hogy a legjobb-e. Ha minden él helyet felveszünk egy-egy irányított élt oda-vissza, akkor egy irányított gráfot kapunk. Megtehetjük ezt a lépést mivel a valóságban full-duplex linkekkel vannak összekötve a rendszer elemei. Ha ezzel csökkenteni tudjuk a feldolgozandó adat mennyiségét, akkor meg is érné ezt a modellt használni.

Az adatközpontokra vonatkozó megállapítások [5.1.], beszűkítik a lehetőségeinket, főleg az utolsó. Azt utolsó feltétel két tervezési filozófiára szűkíti a lehetőségeinket.

1. Az élek lefogása minden switchelési változtatás esetén teljes lefedettséget biztosít, állandó fizikai hozzáférési pontokat használ, nem feltétlenül optimális, a lefogást konfiguráló egységnek kevés adat elég.

2. A konfigurációs egység minden egyes topológiához optimális lefogást tud tervezni, megvalósítani. Ez sok fizikai hozzáférési pontot igényel, és a topológia pontos ismeretét minden idő pillanatban.

Optimális megoldás lenne, az első filozófia használata, optimális lefogással. A második filozófia nehezen megvalósítható, praktikusan túl drága.

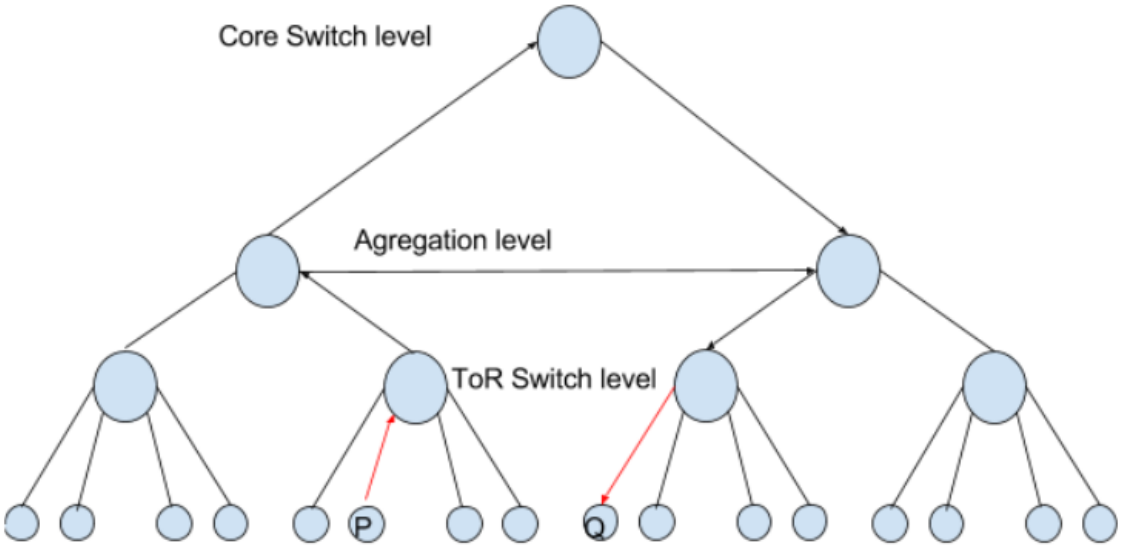
Menger tétele:

Legyenek P és Q egy véges irányított gráf pontjai. Ekkor a P-ből Q-ba vezető éldiszjunkt irányított utak maximális száma megegyezik az összes P-ből Q-ba vezető irányított utat lefogó élek minimális számával.

Minden szerver-szerver éldiszjunkt irányítottot utat lekell fogni, minimális él felhasználásával. Azért nem elég a legrövidebb utakat lefogni, mivel torlódás esetén a switchek dönthetnek sub-optimális út használatáról, avagy az élek súlya folyamatosan változik a switch bufferek megtelése miatt 12. ábra. Műszakilag lehetetlen tudni melyik út éppen a legrövidebb, ezért tényleg lekell fognunk az összes él diszjunkt utat.

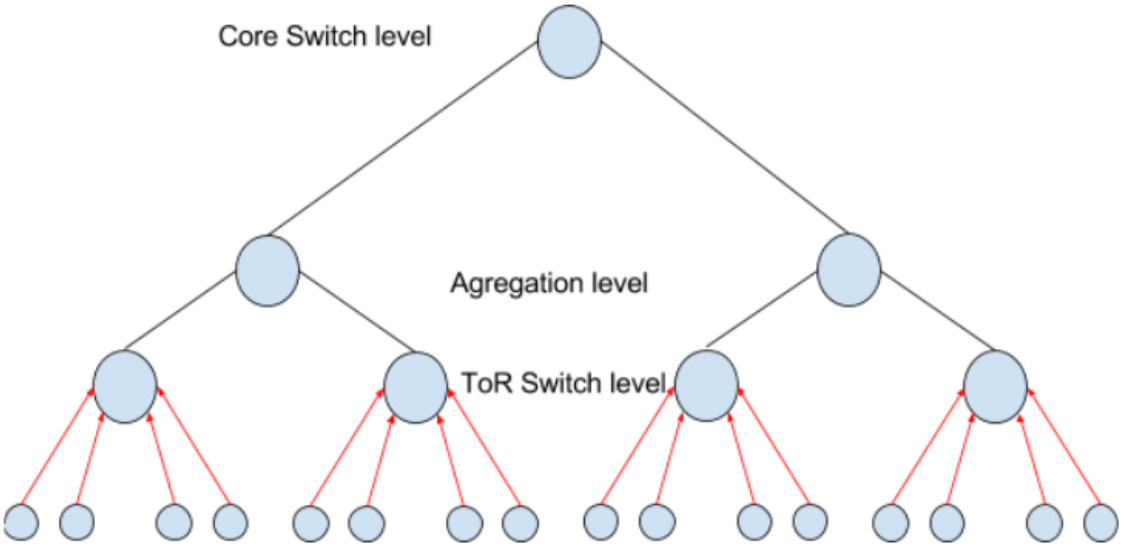
A gráfban szerverek száma legyen n. Legyen P és Q egy-egy tetszőleges szerver pont, ha elfogadjuk a fentebbi DCN jellemzőt (egy virtuális gép egy switchel van összekötve

), akkor látható, egy éldiszjunkt út van minden P és Q közt 12. ábra.



12. ábra P-Q utak

Ezt a szerverek és ToR switchek közti egyszeres összeköttetés garantálja. A P-ToR felfele irányított éllel lefogjuk az minden másik szerverbe tartó összes éldiszjunkt útat. Mohó algoritmus szerint célszerű lenne az összes felfelé irányított szerver-ToR élet lefogni 13. ábra. Másik lehetőség az összes lefele irányított szerver-ToR élet lefogni.



13. ábra Optimális lefogás

Egyszer fogunk le minden szerver-szerver élt, ez nem rossz, de matematikailag nem feltétlenül optimális. Ez az első feltételünket teljesíti mivel a TOR switchek mirrorjain kiadott csomagok MAC-jeivel meg lehet valósítani ezt a lefogást, tudnunk kell a ToR switchek és a szerverek MAC-jeit. A switchelési változtatások nagyrészt felsőbb szinten történnek. Ezt a lefogást matematikailag nem fogom tovább optimalizálni. Innentől áttérek a mérnöki optimalizációra.

Adatközpontokban következő fajta üzenetek fordulnak elő:

-Külső host-szerver üzenet. (érdektelen, belső támadásokra nézve)

-Szerver-külső host üzenet. (érdektelen, belső támadásokra nézve)

-Szerver-szerver üzenet. (érdekes)

-Vezérlő üzenetek, pl. ICMP. (Bármi-bármi) (érdektelen, belső támadásokra)

Vegyük a fentebb említett Szerver-ToR upstream lefogást 13. ábra. Ez tartalmazni fogja az összes Sz-Sz üzenetet egyszer, az összes Sz-Kü üzenetet egyszer, és a vezérlő üzenetek egy részét.

A vezérlő üzenetek a teljes forgalom nagyon kis hányadát teszik ki, és nagyon diverzek, ezért eltekintünk tőlük a további optimalizáció során. Ha eltávolítjuk Sz-Kü üzeneteket, optimális lesz az üzenet halmazunk. Ehhez ismerni kell a core routerek azonosítóit. Ha tudjuk az IP címeiket kifilterezhetjük őket.

Tehát feldolgozás optimalizálásához elég két szűrési feltételt írni a csomagokra:

-Source MAC = DCN szerver MAC

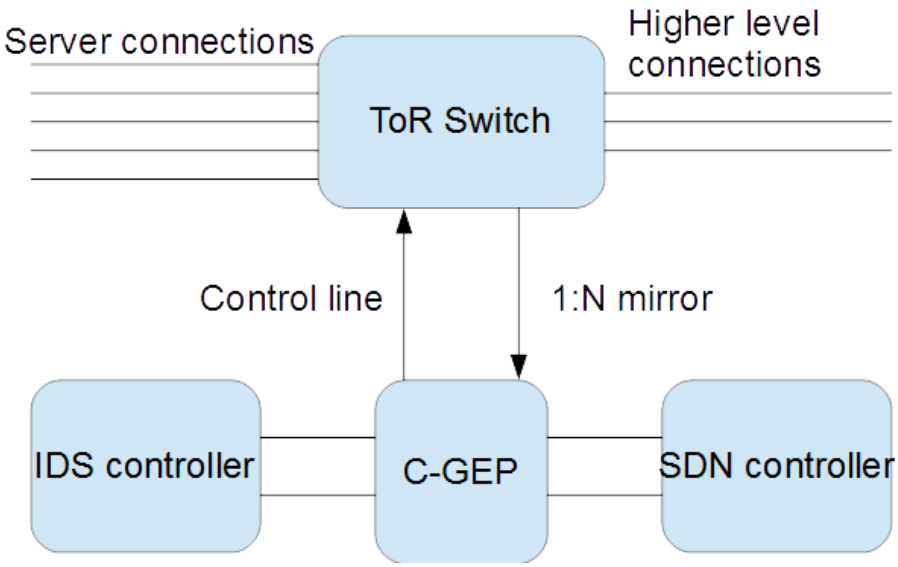
-Destination IP != core router IP

6. A rendszer validálása DDoS UDP flood támadással

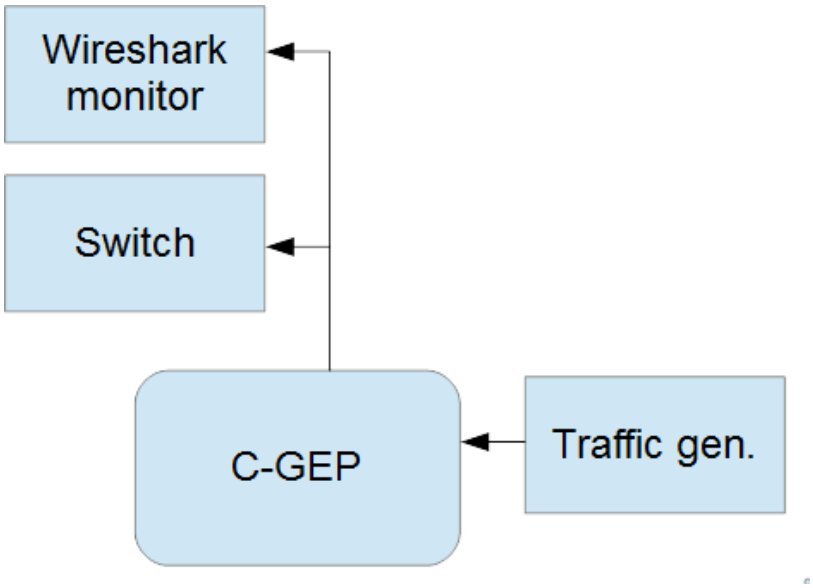
Az UDP flood felhasználása logikus a rendszer kipróbálására és validálására. Egyrészt rendelkezésre áll egy valós, friss támadás (NIIFI Cloud) csomagra pontos, lementett változata. Másrészt egy egyszerű, könnyen detektálható, de mégis nagyon gyakran használt támadási forma ez – tehát életszerű ennek használata a verifikáció részeként. Az UDP floodot könnyebb tisztán hardwaresen detektálni, mint egy Layer 7 támadást. A hardware-ből való detektálhatóság nagyon fontos, mivel a rendszer software IDS komponense nem áll készen tesztelésre.

6.1. A mérés összeállítása

A valós elrendezés helyett 14. ábra**Hiba! A hivatkozási forrás nem található.** a helyettesítő, verifikációs mérési**Hiba! A hivatkozási forrás nem található.** összeállítást használjuk 15. ábra.



14. ábra A rendszer valós elrendezése



15. ábra A mérés összeállítása

Azért ezt az összeállítást használjuk, mert a valós támadás már egy mirroron keresztül kiadott, megcsonkolt adathalmaz formájában van meg.

A rendszer validálásához három dolgot kell:

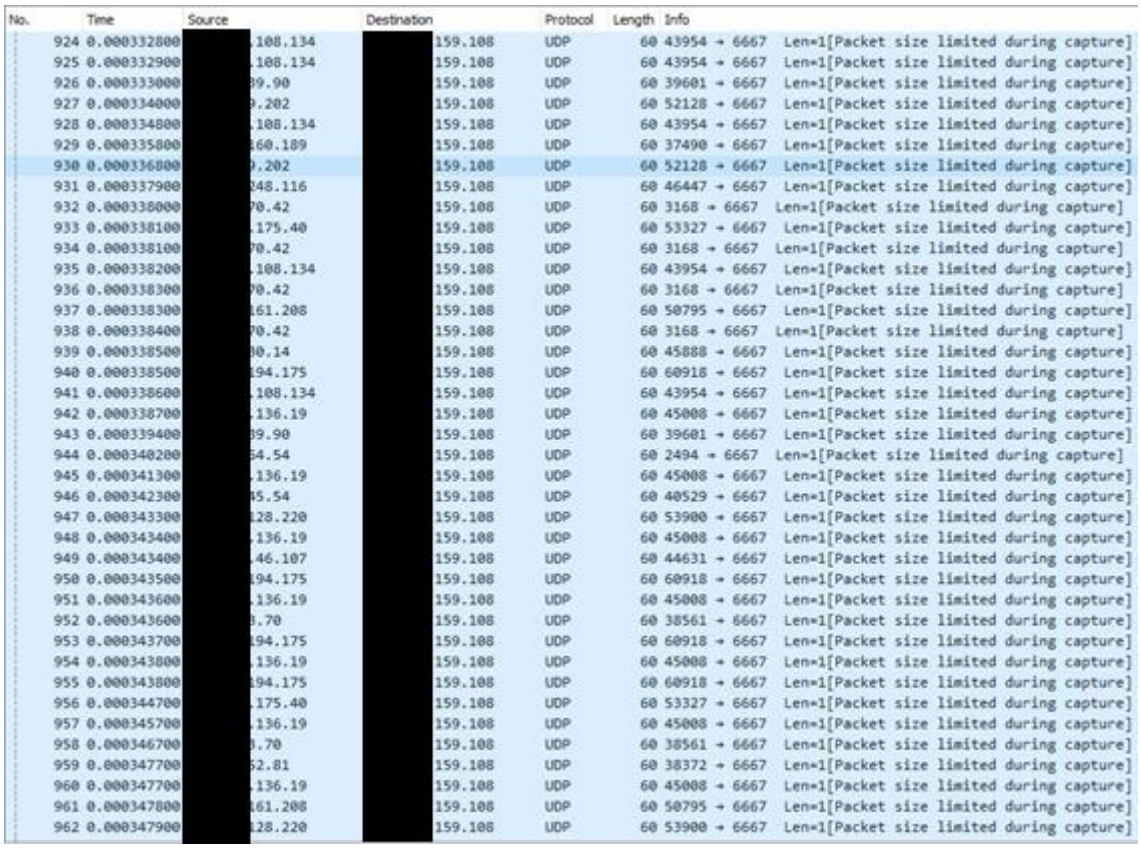
1) A switch mirrorja működik: képes kiadni a forgalmat a megfelelő formában.
2) A C-GEP működik: a forgalom hatására kiadja a kívánt vezérlő üzeneteket.
3) A switch vezérlése működik: a vezérlő üzenetekből képes megfelelő filterezési szabályokat felállítani.

A switch tesztelését sajnos ezen dolgozat keretei között nem mutathatom be. Marad a C-GEP tesztelése.

6.2. A támadás

2016. nyarán a Nemzeti Információs Infrastruktúra (NIIFI) szerver parkját DDoS támadás érte. Az AITIA Zrt. és a BME a NIIFI szerver parkjának egy switch-én elhelyezhette egy saját fejlesztésű megfigyelő rendszerét. A megfigyelő rendszer abban az időben arra volt kalibrálva, hogy nagy méretű forgalom változás esetén elkezdjen rögzíteni. Fizikailag a switch egy mirrorján rögzítettünk, ahol csonkolt csomagokat kaptunk egy 10Gbit/s-s interface-en. A minta, amit feldolgozok a legnagyobb támadás amit elkaptunk, a csúcs értéke 3 Gbit/s támadó forgalom volt.

A támadók 30+ hostot használtak, random portokról azonos hosszú, rövid, értelmetlen UDP üzeneteket küldtek. Az áldozat egy, a NIIFI rendszer részét képező gép (lásd 16. ábra). A forgalom bizonyos részét adatvédelmi okok miatt eltávolítottam.

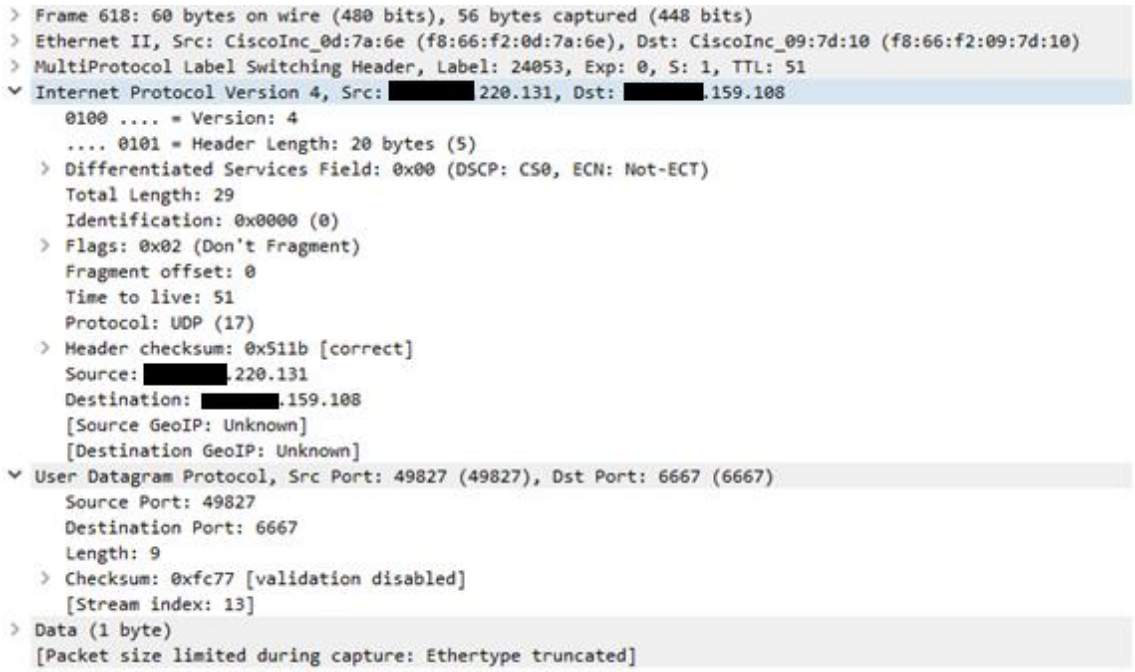


No.	Time	Source	Destination	Protocol	Length	Info
924	0.000332800	[redacted].108.134	[redacted]159.108	UDP	60	43954 → 6667 Len=1[Packet size limited during capture]
925	0.000332900	[redacted].108.134	[redacted]159.108	UDP	60	43954 → 6667 Len=1[Packet size limited during capture]
926	0.000333000	[redacted]89.90	[redacted]159.108	UDP	60	39601 → 6667 Len=1[Packet size limited during capture]
927	0.000334000	[redacted]9.202	[redacted]159.108	UDP	60	52128 → 6667 Len=1[Packet size limited during capture]
928	0.000334800	[redacted].108.134	[redacted]159.108	UDP	60	43954 → 6667 Len=1[Packet size limited during capture]
929	0.000335800	[redacted]160.189	[redacted]159.108	UDP	60	37490 → 6667 Len=1[Packet size limited during capture]
930	0.000336800	[redacted]9.202	[redacted]159.108	UDP	60	52128 → 6667 Len=1[Packet size limited during capture]
931	0.000337900	[redacted]248.116	[redacted]159.108	UDP	60	46447 → 6667 Len=1[Packet size limited during capture]
932	0.000338000	[redacted]70.42	[redacted]159.108	UDP	60	3168 → 6667 Len=1[Packet size limited during capture]
933	0.000338100	[redacted].175.40	[redacted]159.108	UDP	60	53327 → 6667 Len=1[Packet size limited during capture]
934	0.000338100	[redacted]70.42	[redacted]159.108	UDP	60	3168 → 6667 Len=1[Packet size limited during capture]
935	0.000338200	[redacted].108.134	[redacted]159.108	UDP	60	43954 → 6667 Len=1[Packet size limited during capture]
936	0.000338300	[redacted]70.42	[redacted]159.108	UDP	60	3168 → 6667 Len=1[Packet size limited during capture]
937	0.000338300	[redacted]161.208	[redacted]159.108	UDP	60	50795 → 6667 Len=1[Packet size limited during capture]
938	0.000338400	[redacted]70.42	[redacted]159.108	UDP	60	3168 → 6667 Len=1[Packet size limited during capture]
939	0.000338500	[redacted]80.14	[redacted]159.108	UDP	60	45888 → 6667 Len=1[Packet size limited during capture]
940	0.000338500	[redacted]194.175	[redacted]159.108	UDP	60	60918 → 6667 Len=1[Packet size limited during capture]
941	0.000338600	[redacted].108.134	[redacted]159.108	UDP	60	43954 → 6667 Len=1[Packet size limited during capture]
942	0.000338700	[redacted].136.19	[redacted]159.108	UDP	60	45008 → 6667 Len=1[Packet size limited during capture]
943	0.000339400	[redacted]89.90	[redacted]159.108	UDP	60	39601 → 6667 Len=1[Packet size limited during capture]
944	0.000340200	[redacted]54.54	[redacted]159.108	UDP	60	2494 → 6667 Len=1[Packet size limited during capture]
945	0.000341300	[redacted].136.19	[redacted]159.108	UDP	60	45008 → 6667 Len=1[Packet size limited during capture]
946	0.000342300	[redacted]45.54	[redacted]159.108	UDP	60	40529 → 6667 Len=1[Packet size limited during capture]
947	0.000343300	[redacted]128.220	[redacted]159.108	UDP	60	53900 → 6667 Len=1[Packet size limited during capture]
948	0.000343400	[redacted].136.19	[redacted]159.108	UDP	60	45008 → 6667 Len=1[Packet size limited during capture]
949	0.000343400	[redacted].46.107	[redacted]159.108	UDP	60	44631 → 6667 Len=1[Packet size limited during capture]
950	0.000343500	[redacted]194.175	[redacted]159.108	UDP	60	60918 → 6667 Len=1[Packet size limited during capture]
951	0.000343600	[redacted].136.19	[redacted]159.108	UDP	60	45008 → 6667 Len=1[Packet size limited during capture]
952	0.000343600	[redacted]8.70	[redacted]159.108	UDP	60	38561 → 6667 Len=1[Packet size limited during capture]
953	0.000343700	[redacted]194.175	[redacted]159.108	UDP	60	60918 → 6667 Len=1[Packet size limited during capture]
954	0.000343800	[redacted].136.19	[redacted]159.108	UDP	60	45008 → 6667 Len=1[Packet size limited during capture]
955	0.000343800	[redacted]194.175	[redacted]159.108	UDP	60	60918 → 6667 Len=1[Packet size limited during capture]
956	0.000344700	[redacted].175.40	[redacted]159.108	UDP	60	53327 → 6667 Len=1[Packet size limited during capture]
957	0.000345700	[redacted].136.19	[redacted]159.108	UDP	60	45008 → 6667 Len=1[Packet size limited during capture]
958	0.000346700	[redacted]8.70	[redacted]159.108	UDP	60	38561 → 6667 Len=1[Packet size limited during capture]
959	0.000347700	[redacted]62.81	[redacted]159.108	UDP	60	38372 → 6667 Len=1[Packet size limited during capture]
960	0.000347700	[redacted].136.19	[redacted]159.108	UDP	60	45008 → 6667 Len=1[Packet size limited during capture]
961	0.000347800	[redacted]161.208	[redacted]159.108	UDP	60	50795 → 6667 Len=1[Packet size limited during capture]
962	0.000347900	[redacted]128.220	[redacted]159.108	UDP	60	53900 → 6667 Len=1[Packet size limited during capture]

16. ábra Minta a NIIFI elleni támadásból

```
> Frame 1: 60 bytes on wire (480 bits), 56 bytes captured (448 bits)
> Ethernet II, Src: CiscoInc_0d:7a:6e (f8:66:f2:0d:7a:6e), Dst: CiscoInc_09:7d:10 (f8:66:f2:09:7d:10)
> MultiProtocol Label Switching Header, Label: 24053, Exp: 0, S: 1, TTL: 57
v Internet Protocol Version 4, Src: [redacted].128.220, Dst: [redacted].159.108
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 29
    Identification: 0x0000 (0)
  > Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 57
    Protocol: UDP (17)
  > Header checksum: 0x0455 [correct]
    Source: [redacted].128.220
    Destination: [redacted].159.108
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
v User Datagram Protocol, Src Port: 53900 (53900), Dst Port: 6667 (6667)
    Source Port: 53900
    Destination Port: 6667
    Length: 9
  > Checksum: 0xa5c8 [validation disabled]
    [Stream index: 0]
> Data (1 byte)
  [Packet size limited during capture: Ethertype truncated]
```

17. ábra A támadás egy csomagja



18. ábra A támadás egy másik csomagja

A támadók által küldött csomagok nagyon hasonlítanak egymásra (lásd 17. ábra és 18. ábra). Nagy valószínűséggel kijelenthetjük, hogy ténylegesen különböző hostokról jönnek, mivel nagyon sok IP címet használ, és a TTL (time to live) különbözik a csomagokban. A másik lehetőség, hogy spoofolt IP-ket használ, de akkor a TTL megegyezne. A csomagok különbözősége és a több host együtt BOTNET támadásra utal.

6.3. Az UDP flood észlelési eljárás

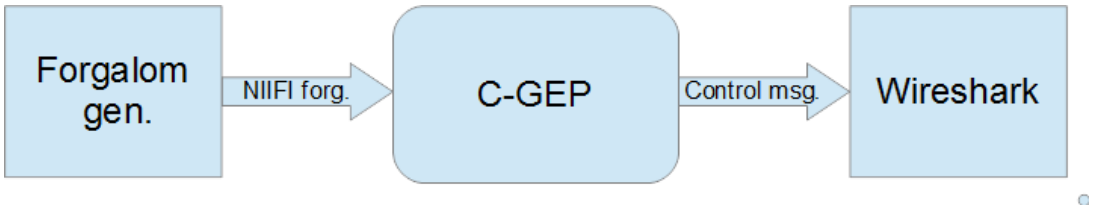
Az FPGA-ba integrált UDP flood attack észlelési módszerünk azon az ökölszabályon alapul, hogy a támadó egy rövid csomagot fog küldeni, erőforrásait optimalizálva. Az UDP flood detektáló hardware ezt használja ki: a csomagokat egymáshoz és ismert mintákhoz hasonlítja [18]. Ezt a módszert szaknyelv „signature analysis”-nek hívja. Ez egy nagyon effektív módszer, mivel nagyon kevés erőforrást használ. A módszer árnyoldala, hogy nem képes randomizált komponenst tartalmazó támadásokat észrevenni (sok esetben).

6.4. A módszer skálázhatósága

Ez a módszer rendkívül jól skálázható, nagyon kevés erőforrást igényel. A C-GEP 100Gbit/s sebességgel érkező csonkolt forgalom feldolgozására is alkalmas ezzel a detektorral, ez 300-400Gbit/s valós forgalomnak felel meg. Ez természetesen függ a támadó csomagok/valódi csomagok aránytól – a NIIFI-s mérések ezt a fent említett arányt hozták.

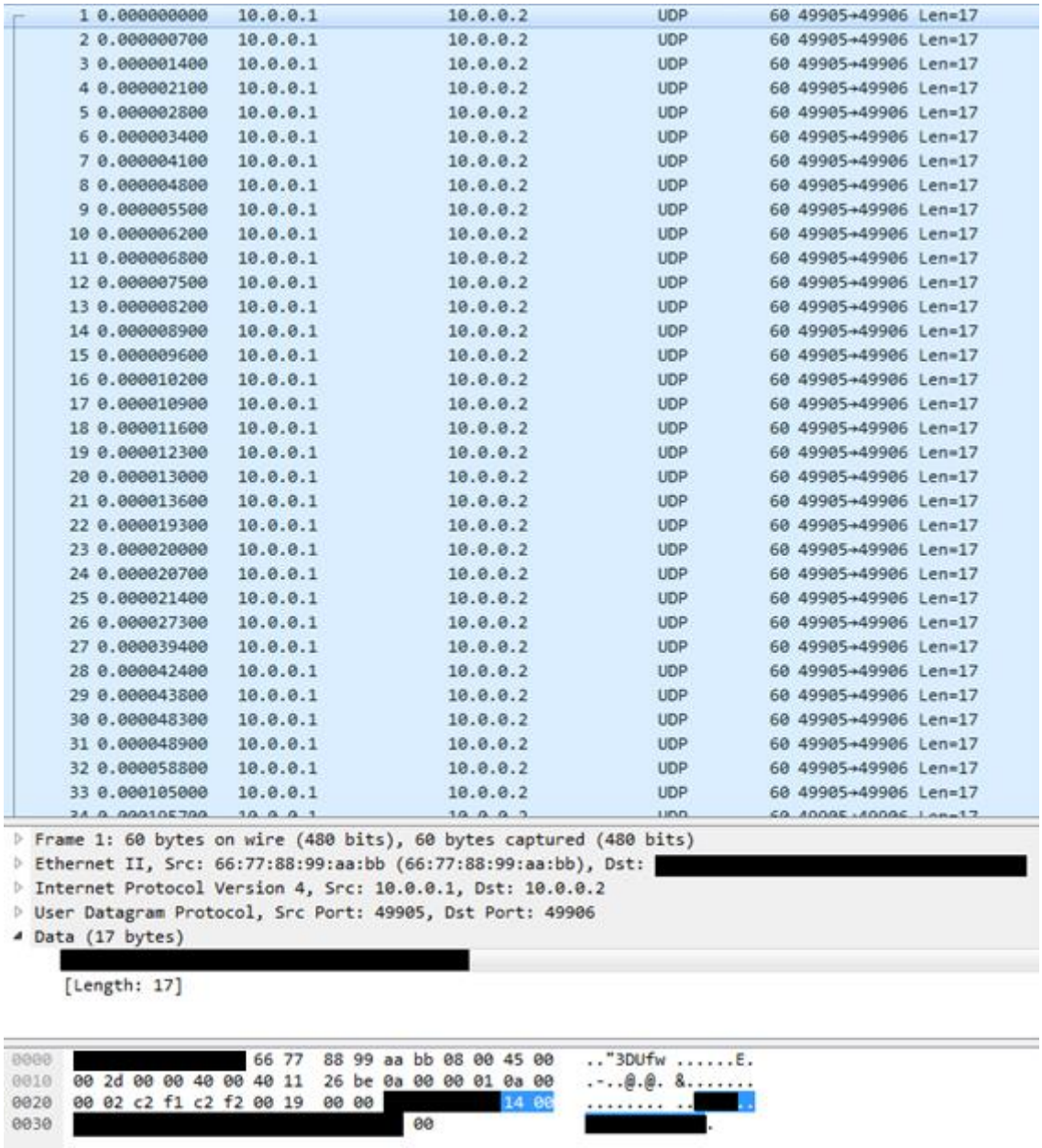
6.5. A teszt lefolyása

A tesztben azt szeretnénk látni, hogy hogyan megjelennek a támadó csomagok a C-GEP bemenetén, az úgy állítja elő és küldi el a switch control üzeneteket (19. ábra).



19. ábra Az UDP-támadás detekciójának mérési elrendezése

A szimulációknak megfelelően meg is történik a behatolás jelzés 20. ábra.



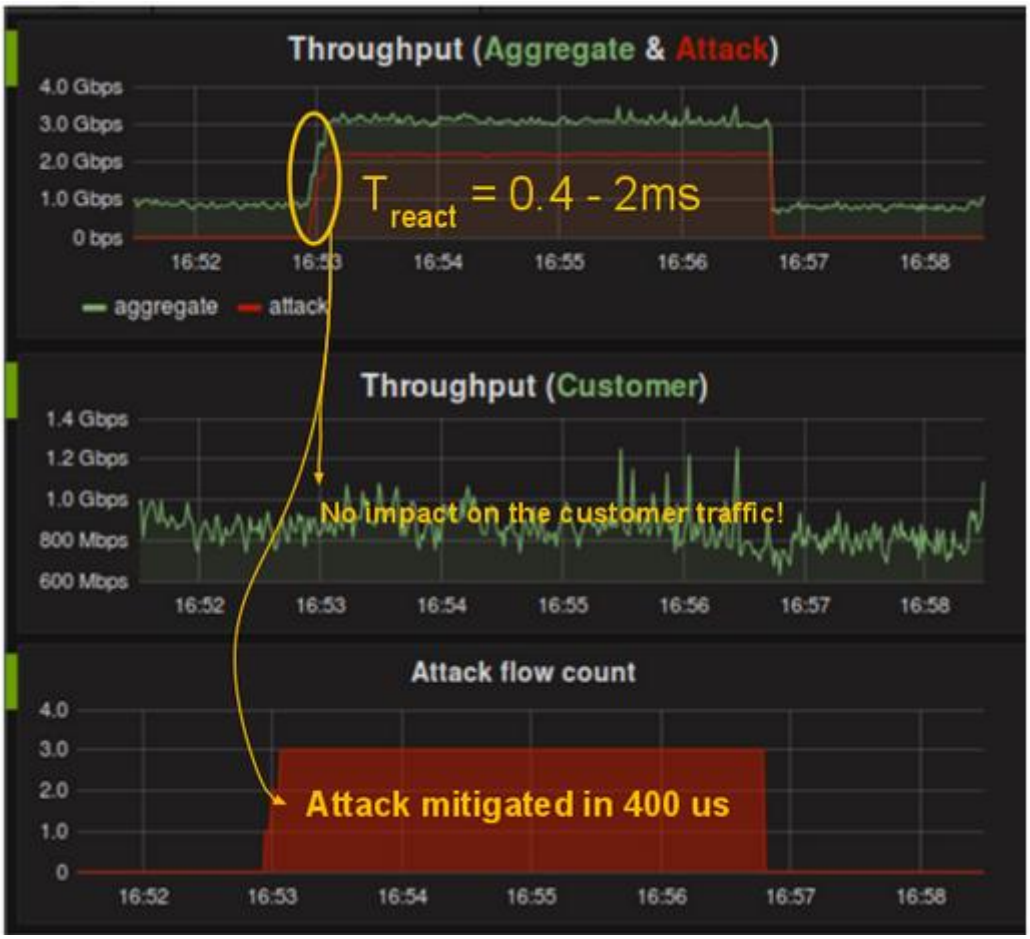
20. ábra A C-GEP kimenete.

A 20. ábra egy bejegyzése, egy a switchnek küldött filterezésre utasítást adó üzenet, az IP-k kézzel kiosztott értékek, a payload tartalmaz

- egy magic kódot (4 byte a payload elején, titoktartási okok miatt törölve)

- egy utasítást, 14 00hexa (IP cím tiltása) és végül
- a tiltani kívánt IP-t (adatvédelmi okok miatt törölve).

A switch MAC címe lett még kitakarva az ábrán. A C-GEP kimenete lassabb, mint bemenete, ezért látjuk az utasításokat szabályos időközönként kijönni: a hálózati interface kapacitása korlátozza a sebességet.



21. ábra Illusztráció a támadás mitigálásához

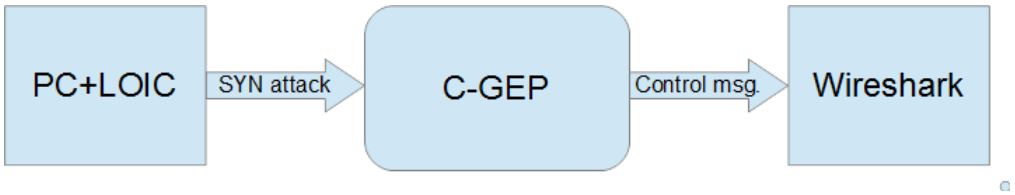
A rendszer reakció sebessége a switch reakció-sebességétől függ elsősorban (21. ábra), ott jelenleg 2ms-ként kerülnek az új szabályok alkalmazásra.

7. Rendszer validálása SYN flood támadással

A rendszer teszteléséhez elengedhetetlen egy másik fajta támadással való tesztelés. A SYN flood támadásra azért esett a választásunk, mert egyszerű software komponens nélkül detektálni, nagyon népszerű támadási forma, könnyen szintetizálható, és a detekciós algoritmus jelentősen különbözik az UDP detekciótól.

7.1. A mérés összeállítása

A tesztelrendezés (22. ábra) itt nagyon hasonló lett volna az UDP floodos tesztéhez.



22. ábra A TCP SYN flood támadás detekciójának mérési összeállítása

A különbség az, hogy a nagy teljesítményű forgalom generátor helyett egy PC-n futó népszerű DoS toolt, a LOIC-t (Low Orbit Ion Cannon [19]) használunk a támadás szintetizálásához.

7.2. A SYN flood észlelési eljárás elmélete

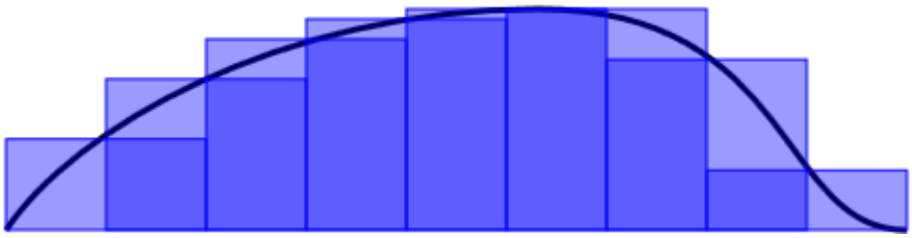
A lehető leguniverzálisabb detektor elkészítése ebben a feladatban a célunk ezért a lehető legkevesebb felvetést használjuk:

1. Sikeres SYN flood támadáshoz fenn kell tartani egy bizonyos nem kicsi SYN csomag/s sebeséget. (Ez a sebesség rendszerről rendszere változik a valóságban)
2. A támadó nem ACK-kolja le a SYNACK üzeneteket.

A második feltevés a SYN flood definíciójából egyértelműen következik. Az első a konkrét számérték kivételével szintén definícióból következik. Az első feltételben előirt SYN sebességet két módon érhetjük el: hirtelen megugrik a célra jutó SYN-ek mennyisége, vagy lassan de folyamatosan növekszik. Célszerű a SYN sebesség differenciáljára fokuszálni, mivel a konkrét sebesség host-ról host-ra jelentősen változik (Példaképpen: egy

banki szerver alkalmazás több SYN üzenetet fog kapni nyugalmi állapotban mint egy kis blog támadás alatt).

-A sebesség változást át kell vinni diszkrét időbe (csak így, diszkrét időben lehet kezelni FPGA-n). Ezt a 23. ábra szemlélteti. A megoldást az $F[k] = \int_{k*t0}^{(k+1)*t0} Fsyn(t)\ dt$ egyenlet alapján tudjuk megtenni, ahol Fsyn(t) a bejövő SYN üzenetek függvényé.



23. ábra Illusztráció a diszkrét időbe való áttéréshez.

A SYN üzenetek számának növekedése nem elégséges egy flood támadás detektálásához. Szükséges látni, hogy a hostok, amik SYN-eket küldik a vélt áldozatnak, küldenek-e másfajta forgalmat is. Ha nem küldenek, akkor feltehetjük, hogy rossz indulatú támadók, mivel nem lehet csak SYN üzenetekkel valós kommunikációt folytatni.

7.3. A SYN flood észlelési eljárás megvalósítása

A SYN flood detektálásához egy kevésbé ismert módszert használunk. Több fázisos hashelésen alapuló forgalom-analízist fogunk alkalmazni.

1. Fázis: A bejövő SYN csomagokra szűrünk, a cél IP-címeket hasheljük. Ezekkel megcímzünk egy memória blokkot, ebbe tároljuk az időegységre jutó SYN-ek számát n időegységre visszamenőleg. A táblázat alapján döntést hozunk, hogy az adott „destination IP hash” támadás alatt áll-e. Amennyiben támadást érzékelünk, a támadott hasht átadjuk a 2. fázisnak.

2. Fázis: A támadás alatt álló IP hash-re szűrünk, csak a SYN és ACK flaggel ellátott TCP üzeneteket tartjuk meg. Ebben a fázisban a source IP-ket hash-eljük. Az alapján mérjük, hogy a „source IP hash”-ekről a támadott ”destination IP hash”-re hány SYN és hány ACK jött. Ha azt látjuk, hogy a SYN-k száma nagyobb mint az ACK-é, és a SYN-k száma nagyobb egy konstans határértéknél, akkor a source és destination IP hash-eket tovább adjuk a 3. fázisnak.

3. Fázis: Ebben a fázisban kikódoljuk a hashekből a támadó IP címét. Kiszűrjük azokat a csomagokat, amiknek az IP hashei megegyeznek a támadó és az áldozat hasheivel, és SYN vagy ACK csomagok. Feljegyezük a source IP-ket, és amelyikre ráillik a támadás profilja SYN > ACK, azt letiltjuk.

7.4. A módszer skálázhatósága

Ez a módszer jelentősen több erőforrást igényel, mint a szignatúra analízis. A legnagyobb fogyasztók a hash táblák, a fejlesztői konszenzus az, hogy 50-100 darab 10 bites hash

tábla férhet a Virtex 6-os chipbe 100G-s MAC mellett. 50-100 hash tábla elég sok mindenre elég, viszont ha meg kell emelni a táblák cím bitjeinek számát gyorsan lecsökken ez a szám az 1. táblázat szerint. Ez a csökkenés azért történik, mert egy hash táblához egyre több BRAM-ot (blokk RAM memória) kell összefogni. A 1. táblázat-ban pirossal és narancs sárgával jelölt soroknál időzítési problémák is fellépnek, praktikusan ebben az FPGA-ban 13 bites a leghosszabb megvalósítható hash.

Hash hossza (bit)	Maximális hash tábla szám (db)
10	100
11	50
12	25
13	12
14	6
15	3

1. táblázat Adott hash hosszhoz megvalósítható táblák száma

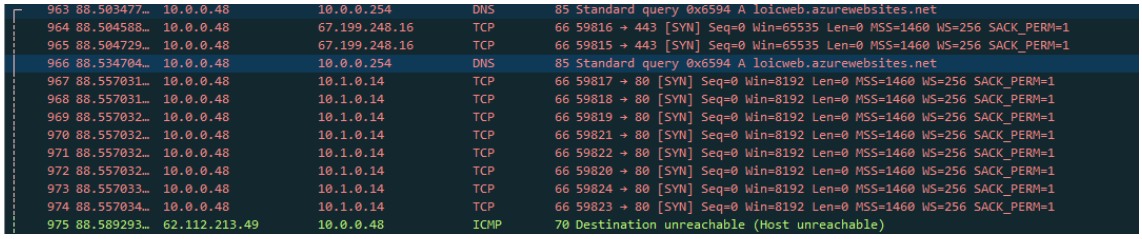
A hash hosszúsága az úgynevezett aliasing (átlapolódás) jelensége miatt lehet kritikus. Tegyük fel, 10 bites hash-et használunk és van 100.000 felhasználó a hálózaton. Átlagosan egy hash címre 100 felhasználó fog jutni, ez azért problematikus mert nagy forgalmazók egy hashen elmaszkolják a kis forgalmú klienseket ért támadásokat. Ezt a jelenséget több módszerrel mitigálhatjuk de mindnek ára van. Ezt a módszert ezen a chipen olyan rendszerekben lehet hatékonyan használni, ahol a felhasználók száma nem éri el ~200.000-t. Adatközpontok belsejében például jól használható.

7.5. A tesztelés menete

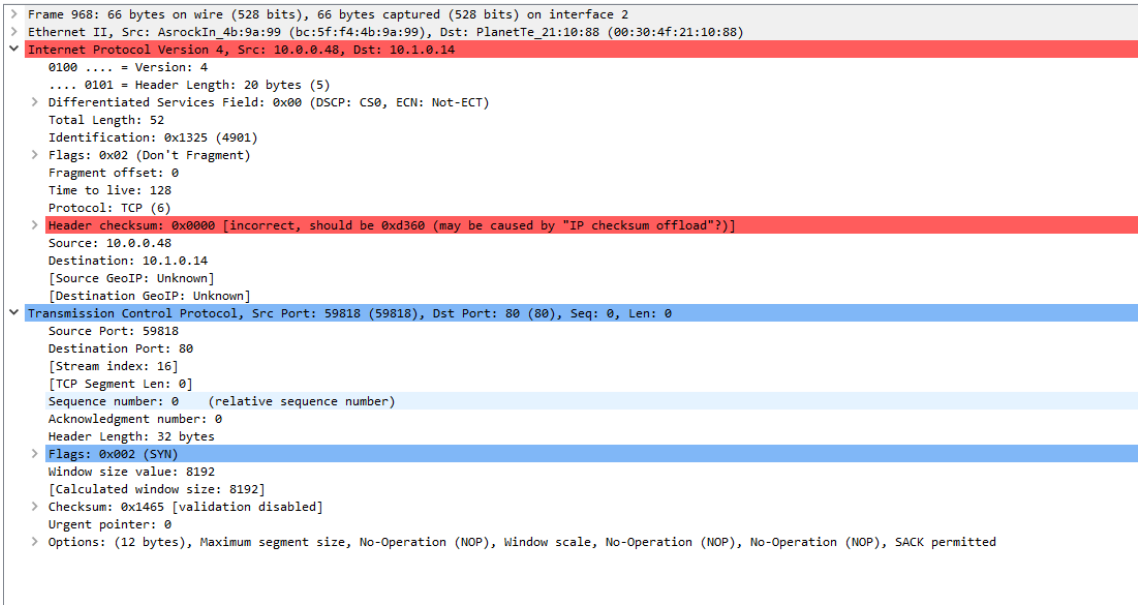
A SYN flood detektor modul fejlesztése sajnos csak a szimulációs tesztek lezárásáig jutott el a dolgozat beadási határidejéig. A modul szimulációját a XILINX ISE [20] segítségével sikeresen elvégeztem, de valós körülmények között még nem vizsgáltam.

A rendszer többi része változatlan az UDP teszthez képest.

A támadást az Anonymus csoport által is használt Low Orbit Ion Cannonnal [19] állítottam elő – lásd 24. ábra, 25. ábra.



24. ábra SYN flood LOIC-kal előállítva



25. ábra LOIC SYN csomag

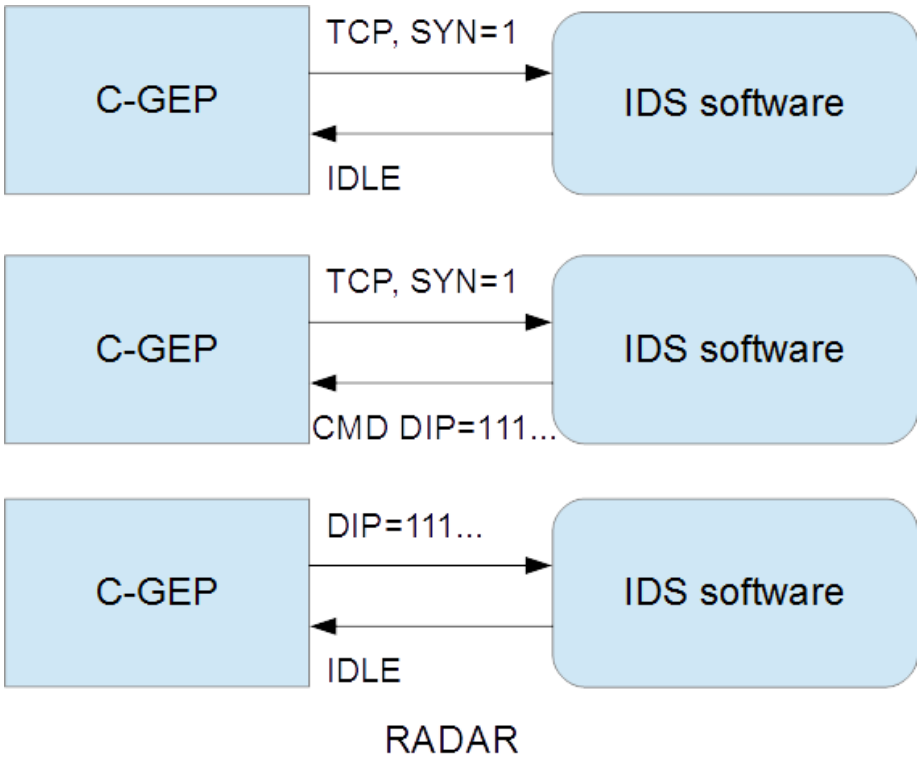
8. Mintavételezés IDS szoftwarenek

FPGA-n nehéz bizonyos támadások elleni védelmet megvalósítani, limitált memória miatt. A fizikai erőforrások erősen korlátozottak ezért célszerű a rendszert software komponenssel is kiegészíteni. A hardware biztosítja a rendszer teljesítményét, a software biztosítja az intelligenciát és flexibilitást. A közelmúltban rengeteg IDS software készült különböző funkcionalitással, de egyik se tud feldolgozni adatot azzal a sebességgel, amire szükség lenne, hogy egy ipari hálózatot kiszolgáljon. Ha IDS softwarét szeretnénk csatlakoztatni a rendszerhez szükséges valami fajta, FPGA alapú csomagszűrési eljárásra. Felhasználhatjuk a gráf elméleti modellt, amit az 5. fejezetben tárgyaltunk, ezzel lényegesen lecsökkenthetjük software által feldolgozandó adat mennyiséget. Ezen felül több fajta filterezési eljárást kidolgozhatunk tovább csökkentve a feldolgozandó adat mennyiséget. Filterezhetünk támadás típus szerint. Ha arra vagyunk kíváncsiak van-e folyamatban HTTP GET támadás akkor, a nem http get csomagokat ki szűrhetjük a softwarenek továbbított adatfolyamból. Filterezhetünk lokalitás szerint, honnan, hova megy a csomag. Ezeket a szűrési módokat idő osztásos módszerrel váltogathatjuk.
Mivel a software egész adatfolyamot nem tudja egy adott időpontban átlátni ezért fontos hogy valamilyen hatás mechanizmussal vissza tudjon jelezni a FPGA-nak hogy milyen adatra van szüksége.

8.1. RADAR működés példa

Példa egy multivektoros támadás feltérképezésére (26. ábra):

1. A FPGA alap rutinja részeként idő osztásos séma szerint váltogatott támadás típusokra filterez, és küld adatott az IDS software felé (TCP, SYN=1).
2. Az IDS észreveszi hogy 111.111.111.111. Host ellen SYN flood támadás van folyamatban. Jelzi a FPGA-nak hogy a SYN csomagok helyet az összes 111.111.111.111.-nek irányuló csomagot továbbítsa. (Destination IP = 111.111.111.111.)
3. Az IDS ebből eldönti ki támadó, feljegyezi összes támadó IP címét. Észreveszi az IDS, ha másfajta támadások is irányulnak ez ellen a host ellen.



26. ábra Példa a RADAR működésére

8.2. Filterezés támadás fajta szerint

A következő felsorolás szűrési feltételeket ad a leggyakoribb DoS támadásokra.

8.2.1. TCP SYN attack

Támadó csomagok: TCP csomagok SYN flaggel. Kizárólag a SYN csomagokból csak különleges esetekben lehet biztosan megmondani hogy van-e támadás (szignatúra analízis), ezért célszerű más TCP üzeneteket küldeni a SYN mellett.

8.2.2. TCP FIN attack

Támadó csomagok: TCP csomagok FIN flaggel. Spoofolt (hamis source IP azonosító) csomagokat használ, FIN csomagok elégségesek a támadás felismeréséhez.

8.2.3. TCP RESET attack

Támadó csomagok: TCP csomagok RST flaggel. Spoofolt csomagokat használ, RST csomagok elégségesek a támadás felismeréséhez.

8.2.4. UDP Fragment attack

Támadó csomagok: Fragmentált IP-UDP csomagok. Elég a támadó típusú csomagokat átküldeni, támadás megállapításához.

8.2.5. UDP Flood attack

Támadó csomagok: UDP csomagok. Elég a támadó csomagokat filterezni, támadás megállapításához.

8.2.6. Slowloris attack

Támadó csomagok: Fragmentált HTTP csomagok. Elég a támadó típusú csomagokat átküldeni, támadás megállapításához.

8.2.7. ICMP flood attack

Támadó csomagok: ICMP (ping) csomagok. Elég a támadó típusú csomagokat átküldeni, támadás megállapításához.

8.2.8. HTTP get attack

Támadó csomagok: HTTP get csomagok. Elég a támadó típusú csomagokat átküldeni, támadás megállapításához .

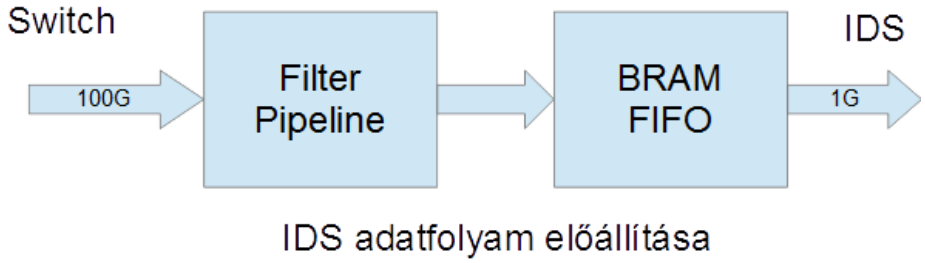
8.2.9. HTTP post attack

Támadó csomagok: HTTP post csomagok. Elég a támadó típusú csomagokat átküldeni, támadás megállapításához.

8.3. Filterezés lokalitás szerint

Mintavételezésnél nem csak a típus szerint lehet szűrni. Érdemes lokalitás alapján működő szűrési eljárást is integrálni a RADAR-ba. A szűrőnek tudnia kell azonosító (IP, MAC, port) alapján filterezni. Az azonosító lehet adott cím, vagy cím tartomány. Ezt érdemes pipeline elrendezésbe fűzni, a pipeline egy elemi művelet végzője egy clock alatt egy komparálást végez el. Egy műveletvégző körülbelül 3ns késleltetést visz a rendszerbe az általunk használt FPGA-val.

8.4. Adatfolyam előállítása az IDS software számára



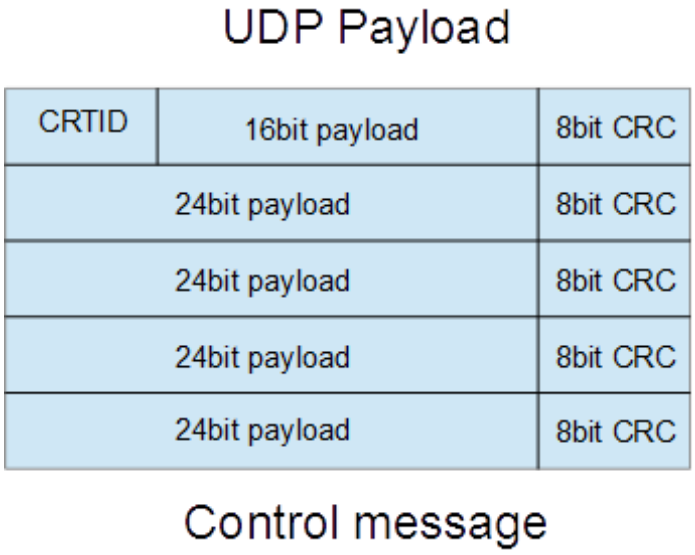
27. ábra IDS adatfolyam előállítása

A fenti blokkvázlat (27. ábra) az IDS softwarebe tartó adatfolyam létrehozását mutatja be. A switchtől érkező adatfolyam először áthalad egy szűrő pipeline-on, ahol eldobjuk a nem kívánt csomagokat. A kiválasztott csomagokat egy FIFO-ba gyűjtjük, majd innen küldjük tovább az IDS felé. Természetesen lehetséges, hogy a kiválasztott csomagok sebessége nagyobb lesz, mint az IDS software kapacitása. A puffer növelésével elkerülhetjük

a tüskék által okozott adatvesztést. Ha hosszabb ideig terheljük túl az IDS software-t, akkor nyilván el kell dobnunk csomagokat. Erről külön frame-ben értesíti a software-t.

8.5. Forgalom leíró keretek

Célszerű az IDS-t tájékoztatni arról is, hogy milyen forgalom van a C-GEP-en amit éppen nem küldünk át, és arról is ha csomagokat dobtunk el az IDS streamjéből. Logikus Ethernet, IPv4, UDP csomagot használni itt is. Ezen üzenetek nagyon nagy mennyiségű információt tárolnak, egy bit hiba hatalmas kárt okozhat a védelmi folyamatokban. Hatalmas különbség, hogy 10 vagy 1034 csomagot dobtunk el az előző 1ms alatt. Ezért kénytelenek vagyunk nagy mennyiségű redundanciát bele építeni az üzenetbe.

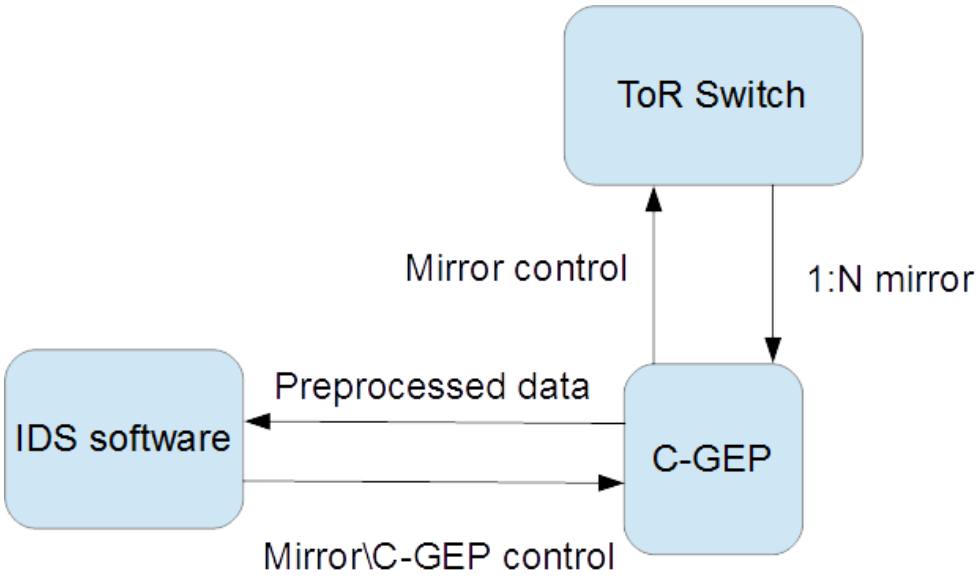


28. ábra Vezérlő üzenet

A fenti 20 byteos üzenet példa a redundáns kontrol üzenetre 28. ábra. Az UDP fejléc portjaival tudjuk jelezni hogy kontrol üzenet jön, ezt ki kell egészíteni egy előre meghatározott CRTID-vel ami jelzi, hogy tényleg kontrol üzenet érkezett.

8.6. A switch vezérlése

A RADAR elvét kiterjeszthetjük a switch vezérlésére is. A mirror nem ideális, azaz nem tud minden egyes csomagot ami switchen áthalad kimásolni. Ha emellett képesek vagyunk vezérelni a mirrort, akkor az előbb bemutatott mechanikát kiterjeszthetjük a switchre is. Azaz a switch mintavételezését vezéreljük az IDS rendszerrel 29. ábra.



29. ábra RADAR switch controllal

Tehát nem csak C-GEP-t szabályozza a software, hanem a mirror mintavételezését is. Ezzel a funkcióval csökkenthetjük a rendszer által lefoglalt switch sávszélességet, jelentősen csökkentve a rendszer alternatíva költségét.

9. Összefoglalás

Napjaink egyik leggyorsabban növekvő cyber fenyegetésévé váltak a DDoS támadások. A dolgozatom célja volt egy újfajta nem invaziv IDS rendszer bemutatása, ami erre a fenyegetésre választ adhat. A dolgozat során megismertem és bemutattam az adatközpontok struktúráját, (D)DoS támadások típusait, a behatolás érzékelés csúcstechnologiáját és problémáit.

A rendszer új ötletek, csúcstechnológiás hardware, és software fúziójából jött létre. Az FPGA alapú hardwarének köszönhető a rendszer csúcstechnologiás sebessége és teljesítménye. A zárt vissza csatolási körön alapuló nem invaziv elrendezésnek köszönhetően, jelenlegi rendszerektől különbözően, nem visz késleltetést a kommunikációba.

A verifikáció során törekedtem minél valósághűbb tesztelésre, bemutattam, hogy a rendszer működik és megállja a helyét ipari körülmények közt is.

Irodalomjegyzék

[1] Paul Sawers, Venturebeat, DYN attack, utolsó hozzáférés 2016.10.21.: http://venturebeat.com/2016/10/21/dyn-dyn-dyn-internet-ddos-attack-back-up/

[2] Wikipedia, Adatközpontok architektúrája, ut. hf. 2016.10.7.: https://en.wikipedia.org/wiki/Data_center_network_architectures

[3] Akamai, State of the internet, 2016 2. negyedév: https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/akamai-q2-2016-state-of-the-internet-security-report.pdf

[4] Incapsula, attack glossary, UDP flood, ut. hf. 2016.10.16.: https://www.incapsula.com/ddos/attack-glossary/udp-flood.html

[5] Incapsula, attack glossary, SYN flood, ut. hf. 2016.10.16.: https://www.incapsula.com/ddos/attack-glossary/syn-flood.html

[6] Incapsula, attack glossary, IP fragmentation, ut. hf. 2016.10.16.: https://www.incapsula.com/ddos/attack-glossary/ip-fragmentation-attack-teardrop.html

[7] A. Khandelwal, N. Jain, S. Kamara, Microsoft, Attacking Data Center Networks from the Inside, 2016.: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/dcn.pdf

[8] Akamai, State of the internet, 2016 1. negyedév: https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/akamai-q1-2016-state-of-the-internet-security-report.pdf

[9] Akamai, State of the internet, 2015 4. negyedév: https://www.akamai.com/us/en/multimedia/documents/report/q4-2015-state-of-the-internet-security-report.pdf

[10] Akamai, State of the internet 2015 3. negyedév: https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/2015-q3-cloud-security-report.pdf

[11] Tamás Tóthfalusi, László Kovács, Péter Orosz and Pál Varga, "100 Gbit/s network monitoring with on-the-fly reconfigurable rules for multi-encapsulated packets", IEEE HPSR, Budapest, 2015., 2016.10.18.: https://www.researchgate.net/publication/281374945_100_Gbits_network_monitoring_with_on-the-fly_reconfigurable_rules_for_multi-encapsulated_packets

[12] Akamai, Client Reputation, ut. hf. 2016.10.18.: https://www.akamai.com/us/en/solutions/products/cloud-security/client-reputation.jsp

[13] Stefanie Hoffman, Fortinet, DDoS, 2013.03.25.: A brief History: https://blog.fortinet.com/2013/03/25/ddos-a-brief-history

[14] Cisco, Cisco.com, Cisco DCN Solution for Network Operations: http://www.cisco.com/networkers/nw00/pres/3301/3301_sec2_DCN.pdf

[15] Cisco, Cisco.com, Cisco Network Solutions for the Telco DCN: SO-NET/SDH OSI Environments, 2007.10.25.: http://www.cisco.com/c/en/us/td/docs/ios/solutions_docs/telco_dcn/tlcsonet.html#wp1306464

[16] Nicole Tripp, toptenreviews.com, DDoS Protection Services Reviews, 2016: http://www.toptenreviews.com/business/internet/best-ddos-protection-services/

[17] Imperva, incapsula.com, Incapsula DDoS protection, ut. hf. 2016.10.17.: https://www.incapsula.com/web-application-ddos-protection-services.html

[18] James Foster, Techtarget, IDS Signature versus anomaly detection, 2016.: http://searchsecurity.techtarget.com/tip/IDS-Signature-versus-anomaly-detection

[19] Wikipédia, Low orbit ion cannon, ut. hf. 2016.10.27.: https://en.wikipedia.org/wiki/Low_Orbit_Ion_Cannon

[20] Xilinx, ISE design suite, ut. hf. 2016.10.27.: http://www.xilinx.com/products/design-tools/ise-design-suite.html

[21] Ankita Manhajan, Oracle, An Introduction to Data Center Network Architectures, 2014.: http://pt.slideshare.net/AnkitaMahajan2/introduction-to-data-center-network-architecture?smtNoRedir=1

[22] FPGA Networking, C-GEP ismertetése. ut. hf. 2016.10.27.: http://www.fpganetworking.com/160616/c_gep/index.html