Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar

Nagy Balázs

DDoS támadások észlelése FPGA-alapú eszközökkel, ezredmásodperces reakció-idővel

Konzulensek

Dr. Varga Pál

2017

TARTALOMJEGYZÉK

Kivonat

A szolgáltatás-megtagadáson alapuló elosztott támadások (Distributed Denial of Service, DDoS) nem számítanak újdonságnak napjainkban, de elhárításuk sajnos még nem megoldott feladat. Erre az elmúlt években több példát láthatunk, mint például a Dyn társaság DNS szerverei elleni támadás. A dolgozatom célja egy gyors reakció-idejű detekciós rendszer bemutatása, amit FPGA-alapú, nagysebességű hálózati platformra terveztem és implementáltam, majd verifikáltam.

Felmerül a kérdés, hogyan lehetne az ipar jelenlegi kihívásaira választ adni. Milyen eszközökkel lehetne hatékonyabbá tenni a tartalomszolgáltatók biztonságát? Ahogy változik a környezet, úgy kell a biztonsági szolgáltatásoknak is adaptálódni. A dolgozatomban bemutatom a jelenleg zajló legfontosabb trendeket, és azt, hogy a jövőben milyen változások várhatóak. A dolgozatom fő témája egy általam tervezett és megvalósított FPGA alapú DDoS detektor, ami ezekre a problémákra keres megoldást.

A dolgozatban bemutatott rendszer olyan hiányosságokra próbál megoldást adni, amire a jelenleg elterjedt rendszerek nem sok sikerrel vállalkoznak. A FPGA különlegessége, hogy egyszerre képes nagy volumenű forgalom valós idejű feldolgozására akárcsak az ASIC, de a tartalma gyorsan változtatható, ha igény van rá. A rendszerem ezredmásodperces reakcióidejével lehetővé teszi, hogy olyan támadásokat detektáljunk és állítsunk meg, amik hagyományos rendszerekkel nem is lettek volna észlelhetőek. A rendszer emellett másodpercenként akár több mint százmillió csomagot képes átvizsgálni, kontextusba helyezni a tartalmukat és döntést hozni ezen információk alapján. Bemutatom, hogy ha egy új támadás megjelenik, milyen kevés időbe kerül az FPGA detektor felkészítése ennek kezelésére – hála a flexibilis és moduláris architektúrának.

Mivel biztonsági tématerületről van szó, ezért kiemelt hangsúlyt fektettem a rendszer tesztelésére. 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

DDoS (Distributed Denial of Service) attacks are not considered novelty these days, but their mitigation is still not a solved issue. There are still many security experts working on making DDoS attacks a thing of the past. In recent years we saw many examples, how these attacks can still cause massive amount of damage, like the attack against DNS servers of DYN Corporation. My paper aims to showcase the intrusion detection system that I designed and implemented for high-speed, FPGA-based networking devices.

The question remains, how one can defend his systems against the next generation of DDoS attacks, what kind of devices could be most efficient in these matters. What kind of devices can take the place of today's constructions? We have to adapt our techniques and tools to battle the ever-changing threats. In my paper, I show the latest trends of this field and what might the future hold for DDoS threats and try to provide a solution to them. The main purpose of my paper is to present the hardware based IDS (intrusion detection system) I designed and implemented.

The hardware-accelerated IDS system described in this paper tries to provide solutions for problems that are extremely challenging for the commercially available solutions. The high packet processing speed and fast on-demand configurability gives FPGAs such a competitive edge that is unique and very different from rigid ASICs and slow and inefficient software. The described IDS system's very short reaction time provides an opportunity to mitigate attacks that are even unnoticeable for most installed IDS. Also, my system can analyze more than a hundred million packets every second, put their information into the context of the traffic and provide intelligent decisions. I showcase how handling of new attacks can be integrated into the system in mere weeks, thanks to the modular and flexible architecture.

Since this is a security-related paper, I put a high quality and quantity of effort into the validation process. The validation process includes the usage of traffic patterns recorded during multiple attacks against NIIFI ((Hungarian) National Information Infrastructure) and the usage for some of the most popular attack tools designed to bring 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 munkatársainak 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 thesis is based on.

1. Bevezetés

Az elmúlt években lényegesen átalakult a DDoS támadások profilja, nem kis részben köszönhető ez az IoT eszközök rohamos terjedésének. A DDoS támadások által elérhető sávszélesség rohamosan nő, nap, mint nap, érdemes a Dyn vállalat elleni 1Tbps támadásra gondolni [1]. Ezek komoly fenyegetést jelentenek a jelenlegi tartalomszolgáltatók működésére, bizonyos esetekben akár a szólásszabadságra is. Ennek legismertebb esete volt a krebsonsecurity.com (egy kiberbiztonsággal foglalkozó blog) elleni – akkor még rekordnak számító – támadás, ami sok napra ellehetetlenítette az oldalt, a támadás Krebs szolgáltatóját az Akamai-t (legnagyobb content delivery network) arra késztette, hogy szerződést bontson Krebs-el, anyagi megfontolásokból [8].

Az IoT eszközök rohamos terjedése, és azok rosszindulatú felhasználása hatalmas átrendeződést hozott a támadások területén, ezzel komoly űrt hozott létre védelem piacán. A dolgozatom ezekre a változásokra próbál egyfajta megoldást keresni.

Az internetes hálózatok fejlődése mellett a monitorozási és védelmi technológiák fejlődése sebességben és technológiai újításokban is elmaradt. Ha most nem fejlesztjük tovább a biztonsági eszközeinket, nagyon komoly árat kell fizetnünk a jövőben. A dolgozatom megalkotásánál nagyon fontos szempont volt egy hasznos és használható rendszer megalkotása.

A TDK-m célja bemutatni egy, részben általam tervezett és megvalósított FPGA alapú csomagfeldolgozó rendszerre írt védelmi rendszert. Egy olyan rendszert, ami képes a DDoS támadások 95%-t ezred másodpercek alatt észlelni és megállítani. Ez azért nagyon fontos mivel egyre gyakoribbak az úgynevezett Hit-and-Run támadások, amik másodperces lefolyásúak és ez alatt is nagyon komoly károkat tudnak okozni. A rendszer alapja az AITIA ZRt. által gyártott C-GEP. A C-GEP egy Xilinx Virtex 6-os FPGA-t tartalmazó csomagfeldolgozó platform. A detektor 100Gbit/s volumetrikus csomagfeldolgozást tesz lehetővé. Az IDS (Intrusion Detector System) tesztelése valós támadások segítségével történt.

A DDoS támadások döntő többségét (95%-át) lefedő 9 különféle támadástípus kivédésére használandó 17, FPGA-alapú algoritmus tervezése, implementációja és verifikációja teljes mértékben a saját munkám; ez alkotja jelen dolgozat gerincét.

2. A DDoS védelmi módszerek ismertetése

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

1. Szignatúra 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 csúcstechnológiát felvonultató rendszerek ezek közül többet is használnak, feltehetően nem publikált saját megoldásokkal kiegészítve.

2.1. Általános védelmi rendszerek

A leggyakoribb védelmi rendszerek általában a hálózat core router(ei)nek tűzfalából és a nagyobb „exploit”-ok kezeléséből (mitigálásából) tevődik össze. A core routerek tűzfalai nagy kapacitású védelmi eszközök, termékenként változó hatékonysággal. A core routerek tűzfala nagy biztonsággal megállítja kívülről érkező támadásokat. Az exploitok mitigálásán azt kell érteni, hogy a hálózatok tervezésékor számoltak különböző gyengeségekkel, amit a hálózat software-es és hardware-es elemei és a szabványok okoznak. Egy jól megtervezett hálózatban is a szabványok alapján dolgoznak, viszont a megvalósításnál igyekeznek csökkenteni a protokollok gyengeségeit, sebezhetőségét. Jellemző példa erre a web szerverek jó konfigurációja.

2.2. 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 le tudja kapcsolni a rendszert. Ha levenné a rendszer üzemeltetője 6 másodpercre a timeoutot (ez még mindig nagyon sok), a támadónak 100-szor több erőforrást kéne használnia. Körültekintő programozással el lehet venni a legtöbb, nem elkötelezett 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.

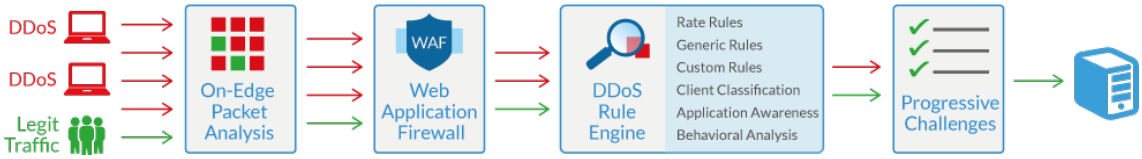
2.3. DDoS védelem elhelyezése szolgáltatóknál

Bizonyos esetekben a felhasználónál kihelyezett védelmi rendszerek nem elégségesek, mivel komoly esélye van annak, hogy a felhasználói hálózatot a támadó telíteni tudja csak az elküldött csomagok mennyiségével. A felhasználóhoz kihelyezett IPS (Intrusion Prevention System, behatolás-megelőző) rendszer sajnos az ilyen hálózat-szaturációt célzó támadásokkal szemben nem hatékony.

Lehetőség van arra, hogy az IPS egységet nem a végfelhasználónál helyezzék el, hanem a szolgáltató rendszerében, ahol sokkal nehezebb telíteni a hálózatott, mivel a szolgáltató oldalán sokkal nagyobb a sávszélesség, mint egy végfelhasználónál.

2.4. Content Delivery Network-ök védelme

Az Imperva üzemelteti a világ legnagyobb, legsikeresebb DoS védelmi szolgáltatását [16], nagyon összetett védelmi rendszerrel dolgoznak. Jelenleg ők képviselik a behatolás-észlelés és -mitigálás csúcsát; a tatalom-szétosztó rendszerekben (CDN, Content Delivery Networks) mindenképp.



1. 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, ezzel a memória és processzor ellen irányuló támadásokat ellehetetlenítik. Ha a kliens IP címét megszerzik a támadók, elárasztathatják a klienst, vagy a hálózati eszközeik kapacitását is kimeríthetik.

Az Imperva védelmi rendszere ezen felül több nagy erejű szerver parkot üzemeltet, hatalmas sávszélességgel. 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. Jelenleg elérhető korszerű DDoS védelmi rendszerek összehasonlítása

Ebben a fejezetben négy gyártó DDoS védelmi különböző megoldásait fogom ismertetni és összehasonlítani azért, hogy áttekintő képet kapjunk az iparról. Ebben a fejezetben tárgyalni fogom a FortiDDoS-t, a Cisco DDoS védelmi eszközeit, a Corsa NSE7000-t, és a FlowMon DDoS Defender-t.

3.1. FortiDDoS

A FortiDDoS egy 100%-ban hardware-es megoldásokon alapuló IPS rendszer. A rendszer egyetlen dobozba van integrálva.

> *„Only Fortinet uses a 100% SPU approach to its DDoS products without the performance compromises of a CPU or CPU/ASIC hybrid system. The SPU-TP2 transaction processors inspect 100% of both inbound and outbound Layer 3, 4 and 7 traffic, resulting in the fastest detection and mitigation, and the lowest latency in the industry."*

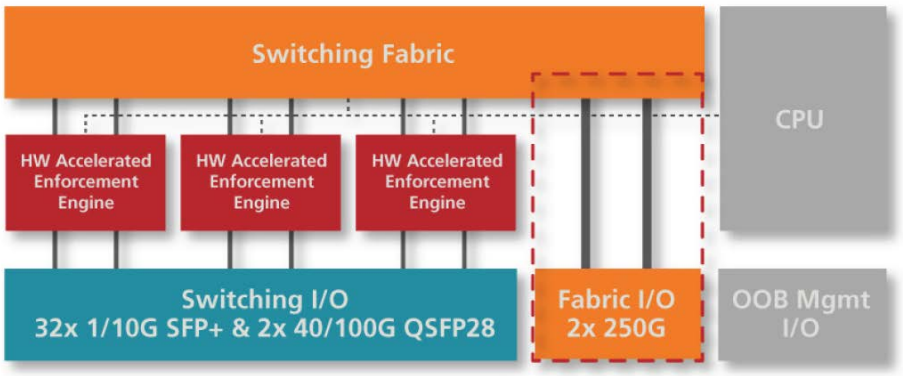
A fenti idézet a FortiDDoS [30] adatlapjából van, azért emeltem ki, mivel nagyon jól megfogalmazza az eszköz különlegességét. A FortiDDoS detekciós része 100%-ban ASIC alapú és 100% viselkedési analízist alkalmaz. Tehát nem adatbázisok alapján dönt, hanem tisztán a forgalom alapján [30]. A FortiDDoS teljes leírása megtalálható a cég honlapján, a FortiDDoS Handbook-ban [31].

3.2. Cisco NCS-5500

A Cisco NCS-5500 a Cisco legkorszerűbb, legerősebb router-e. Maga a router nem tartalmaz IPS rendszert, viszont a route processor modulon futó CISCO iOS-ra lehet telepíteni különböző IDS rendszereket, amik integrálódnak a routing rendszerek effektív használatával [32]. Ebben a rendszerben annyi detekciós potenciál van, amennyi el tud futni a routing mellett a CPU-n. Sajnos itt nem lehet pontos adatokat mondani az IDS-ről, mivel teljesen software-es megoldás, viszont ez a fajta (edge-router-re telepített) megoldás nagyon jellemző még ma is, ezért mindenképp érdemes megemlíteni.

3.3. Corsa NSE7000

A Corsa NSE7000 a kakukktojás, mivel az NSE nem tartalmaz IDS rendszert: kizárólag a támadás megakadályozására alkalmas.



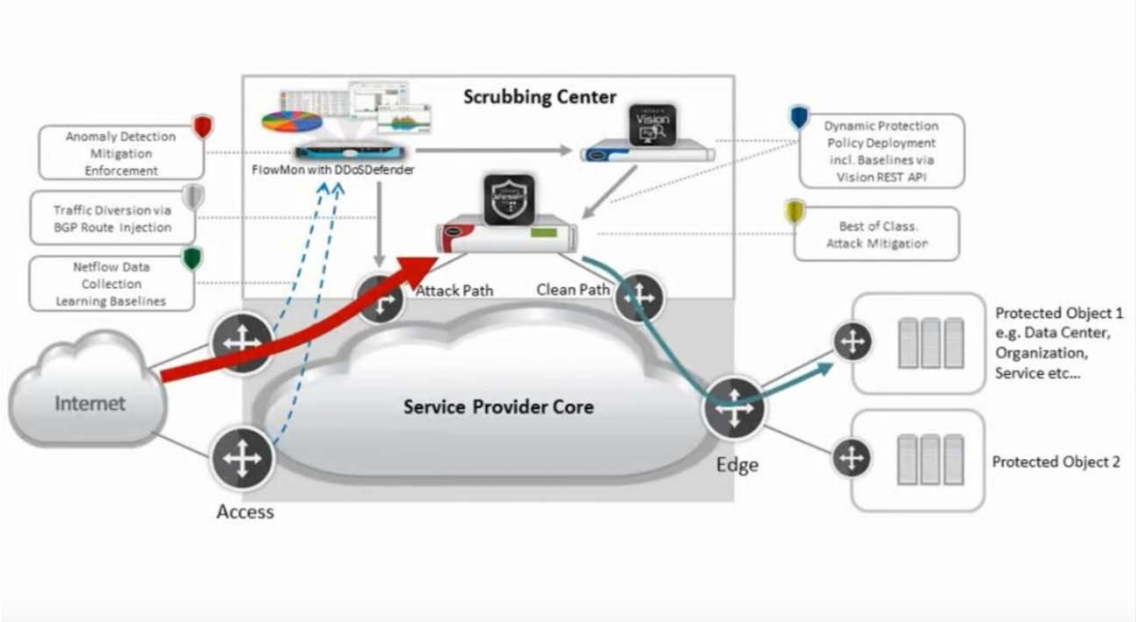
2. ábra Az NSE7000 blokk vázlata [28]

A 2. ábra mutatja be a NSE blokk vázlatát [28]. A képen látható, hogy NSE7000 valójában egy switch, amibe szűrök vannak beiktatva a switching fabric elé.

Azért emeltem ki a NSE7000-t a többi beavatkozó szerv közül, mivel sokkal hatékonyabban végzi a munkáját, mint a hasonló célú eszközök [29]. A biztonsági beavatkozást teljesen elválasztja a routing-tól, ami szolgáltatás oldalán nagy előnyt jelent mivel a „mission team" (routing, content szolgáltatás) és a „security team" egymástól függetlenül működhet.

3.4. FlowMon DDoS Defender

A FlowMon IPS rendszerének több lehetséges üzemeltetési topológiája van, itt csak a Radware integrált topológiát fogom ismertetni.



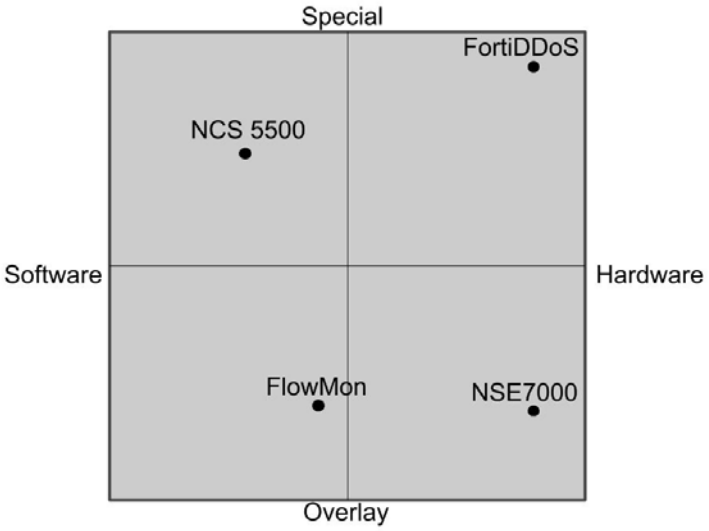
3. ábra FlowMon, Radware integrált hálózat [26]

A 3. ábra mutatja be a rendszer hatás-mechanizmusát [26]. A hálózat edge-router-ei és a FlowMon Probe-ok [27] áteresztik a beérkező csomagokat és feldolgozzák őket, az így elkészült statisztikákat tovább küldik a FlowMon DDoS Defendernek, ami kiértékeli a

statisztikákat és döntést hoz. Fontos megemlíteni, hogy ez a rendszer nincs teljesen automatizálva, operátori megerősítésére vár, csak utána avatkozik közbe. Amint az operátor megerősítette, hogy beavatkozásra van szükség, a Defender átirányítja a DDoS-sal szennyezett forgalmat a hálózaton a Scrubbing Center-be, ahol a szennyezett forgalmat megtisztítják a DDoS-tól.

Ennek a centralizált elrendezésnek komoly előnye nagy hálózatoknál, hogy egy IDS átlátja az egész hálózatot, valamint az IDS-nek nem kell rendelkeznie hatalmas forgalom feldolgozó képességgel. Ezekért az előnyökért nagyon komoly árat kell fizetni. A rendszer nagyon lassú az elosztott rendszerekhez képest. A DDoS mitigáció a hálózat belsejében történik, ami Giga-Tera flood-oknál feleslegesen szaturálja rendszert. A hálózat router-einek képesnek kell előállítani a flow statisztikákat, ami nem egy alap funkció a legtöbb router-nél.

3.5. A rendszerek összehasonlítása



4. ábra Rendszerek összehasonlítása

A rendszereket összehasonlítva látható, hogy azok a védelmi megoldások egy jellegzetes felbontását elég jól lefedik (4. ábra). Az 1. táblázat a kiemelt védelmi rendszerek néhány fontosabb tulajdonságait foglalja össze.

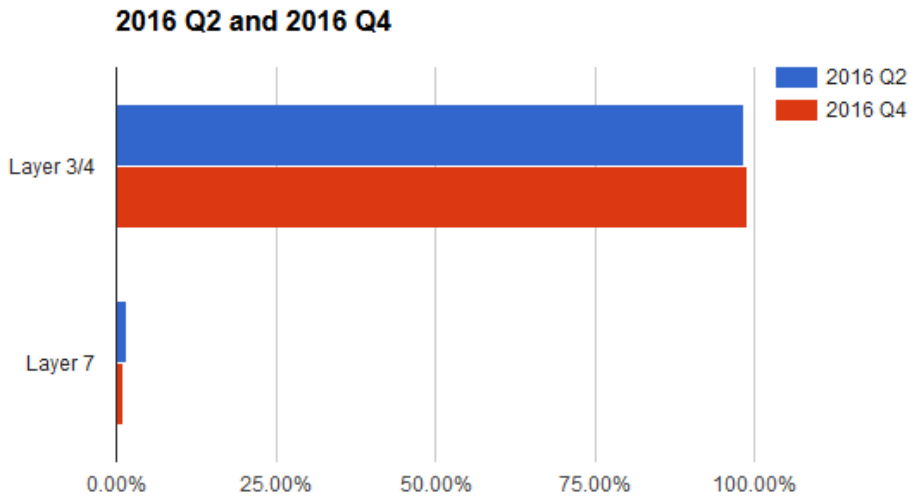
1. táblázat – Kommerciális tűzfalak néhány fontos sajátosságának összehasonlítása

	FortiDDoS	*NCS5500*	*NSE 7000*	*DDoS Def. (FlowMon)*
Reakció idő	<2s	~60s, változó -	<1s	lassú, percek
Architektúra	ASIC	CPU	ASIC	ASIC+CPU
Maximális teljesítmény	48Gbps	változó	200Gbps	36Tbps
Layer 3,4	Igen	Igen	Nem	Igen
Layer 7	Igen	Igen	Nem	Igen
Mitigáció helye	Pre-Edge	Edge-router	Pre-Edge	Hálózat belsejében
Csomag feldolgozási képesség	Veszteség mentes	Veszteség növekszik a komplexitással és a teljesítménnyel	Veszteség mentes	Veszteséges

4. 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 nagyméretű 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 módszert [13]. A DDoS-t mint módszert használják politikai aktivizmusra, zsarolásra és bosszúra is. Az elmúlt években ugrásszerű növekedésen estek át a DDoS támadástípusok erőség és gyakoriság tekintetében is. 2015 és 2016 között jelentős, 139 százalékos növekedés volt erősségben [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 reguláris klienseit. A DDoS támadásokat a szakirodalom két nagy csoportra osztja: (i) Layer-3/4 és (ii) Layer 7. A layer 3/4 dominálják jelenleg a DDoS támadások piacát 5. ábra.



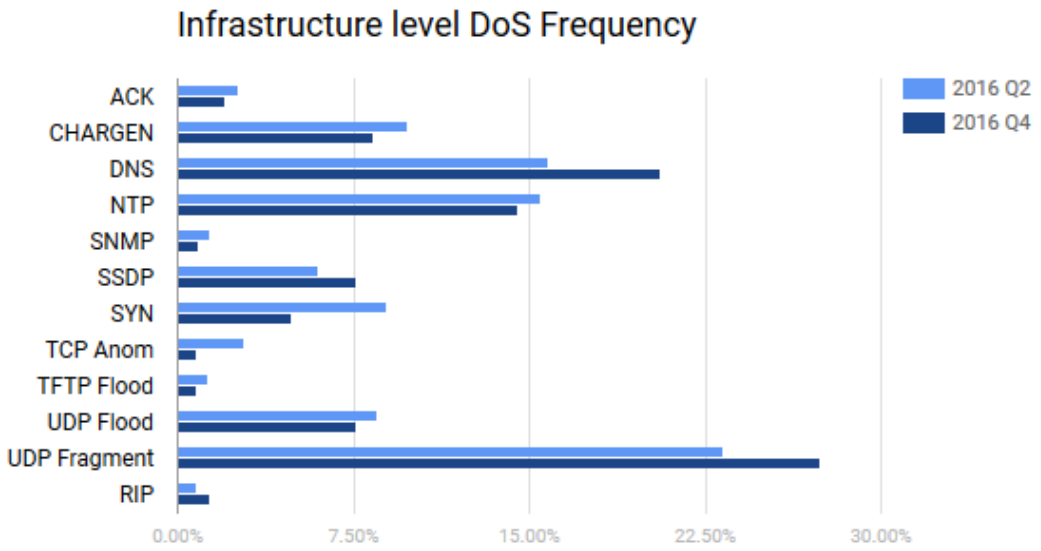
5. ábra DDoS támadások layerek közti eloszlása [2][3]

4.1. Layer-3/4 támadások

A Layer-3/4 támadások a legegyszerűbbek és a leggyakoribbak; a 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 túlterhelésére fókuszál. Leggyakoribb fajtái: UDP flood [4], SYN flood [5], UDP fragmentation [6], és sokféle, erősítésen alapuló támadás, mint az NTP, vagy a DNS [6]. Ezek a támadások csomagszinten primitívek, de nagyon hatékonyak tudnak lenni, ha jól

használják. Az elmúlt évek egyik trendje ezen támadások gyakoriságának növekedése [37].

4.1.1. Layer-3/4 támadások eloszlása



6. ábra DDoS támadások gyakorisága és változása 2016-ban [2][3]

A 6. ábra az Akamai 2016 Q2-s kutatása alapján készült. Ez a „nagy” internet támadásfajtáit foglalja össze. Látható, hogy az UDP alapú flood támadások teszik ki az összes támadás nagy részét, és a TCP alapú támadások népszerűsége csökken. A dolgozatom a 6. ábrán megjelölt 12 támadás közül csak a kilenc leggyakoribbal foglalkozik.

4.1.2. 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 a reguláris flow-knak ne jusson operatív memória [9][33]. Ez a támadás az aszimmetrikus hadviselés elvét használja: egy egységnyi támadó oldali erőforrás több egységnyi áldozat oldali erőforrást használ. Sajnos e tulajdonságok miatt nincs úgynevezett „ezüstgolyó” a SYN flood ellen. Általában szerver oldali megelőző tervezéssel igyekeznek kezelni a támadás hatásait. Mivel ez az egyik legrégebbi támadásforma ezért számtalan detekciós metódus készült rá az évek során. A syn flood népszerűsége csökkenő trendet mutat.

4.1.3. UDP flood

Az UDP flood egy negatívan definiált kategória, tehát minden olyan támadás ami UDP alapú és nem tartozik egyik más fajtába se. Legtipikusabb megvalósítása; a támadó véletlenszerűen kiválasztott portokra küld UDP csomagokat [34]. Az áldozat operációs rendszere megnézi, hogy fut-e olyan applikáció, ami azon a port-on figyel – és ha nincs (általában nincs), akkor egy ICMP üzenetet küld, amiben jelzi: nincs alkalmazás az adott port-on. 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. Viszont a pontos specifikáció hiányában, nehezebb detektálni, mint más jobban specifikált támadásformákat. Valószínűleg rossz hatásfokának köszönhetően csökken ezen támadások aránya.

4.1.4. UDP fragmentation

A leggyakoribb Layer 3,4 támadás [3],[8]. Azt az IPv4 funkciót 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 áldozat-oldalon próbáljak kezelni a támadás hatásait. Ennek a támadásnak a népszerűsége töretlenül növekszik, mivel nagyon hatékony direkt támadás.

4.1.5. NTP

Az NTP visszavert, erősített támadás az egyik leggyakoribb DDoS támadás. A leggyakoribb formája spoofolást igényel. A támadó(k) NTP MON_GETLIST küldenek NTP szervereknek, de a támadók saját IP címük helyet az célpont IP címét írják a source IP mezőbe. A MON_GETLIST parancs lekérdezi az utolsó hatszáz látogató IP címét. Ennek köszönhetően a támadók saját sávszélességüket akár meg is ezerszerezhetik [7][35]. A támadás az áldozat sávszélességét akarja elfogyasztani. A spoofolás és a visszaverés elrejti a támadók valódi kilétét ezért a támadás megakadályozása nem történhet IP címek letiltásával. A támadás megszakítása az áldozat IP címére érkező NTP csomagok megszűrésével történhet. Ez egy nagyon egyszerűen detektálható támadás, de hatásfokának köszönhetően még mindig nagyon népszerű.

4.1.6. DNS

A DNS visszavert, erősített támadás az egyik leggyakoribb DDoS támadás. Szintén egy spoof-olást igénylő támadásfajta. A támadók DNS lekérdezéseket hajtanak végre, amihez az áldozat IP címét kölcsönzik. A DNS szerverek nagy száma és nagy sávszélessége és lekérdezés nagy erősítése teszi az egyik legnépszerűbb támadás fajtává. A DNS nem olyan hatékony, mint az NTP, de könnyebb megvalósítani [36]. Ez a támadásforma az áldozat sávszélességét próbálja felhasználni.

4.1.7. CHARGEN

A CHARGEN (Character Generator Protocol) az Intenet Protocol Suite egy eleme, az RFC 864 [14] definiálja. Ez egy régi, ritkán használt protokoll. A megvalósítása lehet TCP vagy UDP alapú, de DDoS támadásra csak az UDP alapút használják. A protokoll lényege, hogy a megfelelő (19-es) porton beérkező csomagra egy random (0-512) karakter hosszú random választ küld. Látható ez a protokoll is használható DDoS támadásra spoof-olás segítségével [37]. Ez a támadási forma az elmúlt években lett népszerű, ahogy hackerek újra felfedezték ezt a részben elfelejtett protokollt.

4.1.8. SSDP

Az SSDP (Simple Service Discovery Protocol) egy elosztott szolgáltatás-felderítő protokoll. Ezt a protokollt szintén lehet erősítőként használni DDoS támadásokhoz. Az 1900-as porton küldött üzenetekre választ fog küldeni az SSDP eszköz. A CHARGEN-hez hasonlóan ezt a régi, nem biztonságos protokollt is mostanában, 2014-ben fedezték fel újra [7]. Mivel a hatásfoka nem olyan jó, mint a DNS-é vagy az NTP-é ezért valószínűleg nem fog mértékadó támadási formává válni, viszont növekedése várható, amíg a védelmi rendszereket nem készítik fel rá.

4.1.9. ACK flood

Az ACK flood a SYN flood-hoz hasonlóan egy direkt támadási forma, ami a TCP protokollt használja. A hatásmechanizmusa sokkal gyengébb, mint a SYN flood-é, mivel egy csomag csak egy memória-ellenőrzést okoz. Viszont a SYN floodal ellentétben a legtöbb védelmi rendszer nincs felkészítve az ACK flood-ra olyan szinten, mint a SYN flood-ra. Ezt a támadásformát legtöbbször egyszerűbb védelmi rendszerek megkerülésére használják. A hatásfoka az UDP flood-hoz hasonló.

4.1.10. RiPv1

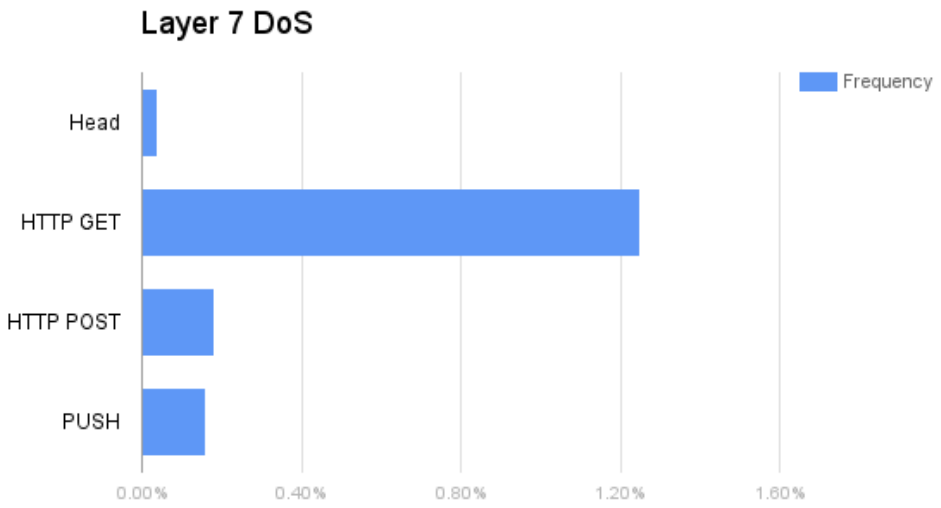
A RiPv1 (Routing Distance Vector Protocol) az egyik első (1988) távolság vektor alapú hálózat feltérképező protokoll. Ahogy a legtöbb, abból a korszakból származó protokollnál, itt sem figyeltek a biztonságra túlságosan. Az SSDP-hez hasonlóan ezt a protokollt is lehet erősítőként használni támadásokhoz. Ezzel a támadás-formával is több mint százszoros erősítést lehet elérni és sok védelmi rendszer számára teljesen ismeretlen, ezért komoly növekedésre lehet számítani az elkövetkező néhány évben.

4.2. 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 hardware-t igényelnek, viszont sokkal nehezebb őket implementálni, mivel itt applikáció-specifikus gyenge pontokat (exploit-okat) kell keresni. 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].

4.2.1. Layer 7 támadások gyakorisága



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

A 7. ábra bemutatja a leggyakoribb layer 7 támadások előfordulási rátáját, ahol a százalékos alap az összes DDoS támadás előfordulása.

4.2.2. 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-igényes lekérdezéseket használják. Ehhez botneteket használnak, hogy a volumetrikus szűrésen ne bukjanak le.
2. Elküld egy legitim GET/POST header-t 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.

4.3. DDoS támadások forrásai

A DDoS támadások leggyakoribb forrásai malware-rel megfertőzött Internet eléréssel rendelkező eszközök. Ezek lehetnek személyi számítógépek, okostelefonok, és egyre gyakrabban IoT eszközök. Ezek az eszközök legtöbb esetben nem rendelkeznek komoly

hálózati és számítási kapacitással, ezért nagy támadásokhoz több ilyen meghackelt eszközt kötnek egy botnet-be.

A jelenlegi legnépszerűbb botnet software az úgy nevezett Mirai botnet, melynek kódját nemrég publikálta [15] az eredeti készítője és azóta több álváltozata jelent meg. A Mirai botnet általában felelőtlenül megtervezett IoT eszközökre vadászik a neten. Legtöbb esetben a hackelést kis körültekintéssel el lehetne kerülni, például nem lenne szabad admin/888888 username/password kombinációt benne hagyni a kódban [21].

4.4. DDoS trendek

A jelenlegi trendeket három csoportba lehet osztani.

Talán a legfontosabb DDoS trend napjainkban a támadások erősségének, sávszélességének növekedése, ezen felül a brute-force layer 3,4 támadások arányának növekedése. Ez a két trend direkt következménye a rosszul megtervezett IoT eszközök terjedésének.

A másik trend csoport a legacy protokollokon alapuló erősített, visszavert támadások térnyerése, ebbe a csoportba tartozik a CHARGEN, RIP, SSDP. Ez a trend a védelmi rendszerek hiányosságait aknázza ki. Ez a trend már sok éve jelen van.

A harmadik trend csoport a védelmi rendszerek lassú reakciós idejének kihasználása. Azaz sok rövid támadást intéznek a célpont ellen, régebbi biztonsági megoldások ezeket nem is észlelik. Ha sikerül észlelni ezeket a gyors lefolyású (Hit-and-Run) támadásokat, akkor ember komponens teherbírását aknázzák ki, azaz azt, hogy a legtöbb megoldásban jelenleg is ott van egy ember, aki döntést hoz, akinek több ezer támadás bejegyzés kerül az interface-re. Ezt a trendet több kiberbiztonsági szakértő leírta már [40][2].

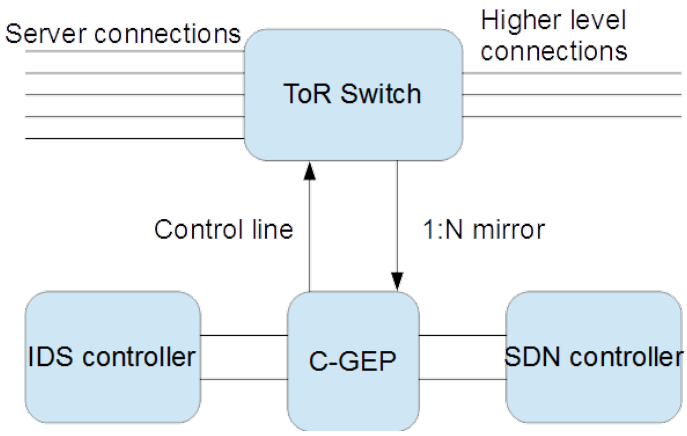
5. Az FPGA alapú védelmi rendszer és környezete

A dolgozat újdonsága, hogy egy aktív visszacsatolási kör topológiába elhelyezett eszközök segítségével készült. Ez egy egyszerűen telepíthető és üzemeltethető rendszer. Az alapötlete az, hogy egy switchen, vagy switch rendszeren átfolyó adatot átadjuk egy feldolgozó és döntéshozó rendszernek, a switch mirror port-ján keresztül.

Az FPGA alapú platformnak köszönhetően az adatfeldolgozás sebessége ezredmásodperces 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. Switch-enként egy linket szabaddá kell tenni a mirror-nak. Hasonló funkcionalitású rendszerek installálása, drága switchek/routerek behelyezésével, vagy a hálózat teljes újratervezésével járna.

5.1. A rendszer felépítése

A switchek a mirror funkciót használva lemásolják a switch forgalmát, ezt a forgalmat feldolgozás céljából átadják a C-GEP-nek. Természetesen a mirroron lemásolt forgalmat megcsonkítva adjuk át a feldolgozónak, a csomagok payload részét levágjuk. A csonkolás azért előnyös, mert az adatok döntő többsége egyébként is titkosított (tehát feldolgozásra alkalmatlan), viszont így megnöveljük a mirroron átvihető csomag mennyiséget. A C-GEP a 8. ábra szerinti elrendezésben feldolgozza ezt a forgalmat és kiszolgálja az IDS software-t, és az SDN controllert.



8. ábra A védelmi rendszer blokkvázlata

5.2. A rendszer időszerűsége

A múltban egy ilyen topológia megvalósíthatatlan lett volna. A kommersz switchek mirror funkciója nem volt képes egy ilyen feladat ellátására, mirrorok ezekben a switchekben nagyon komoly kompromisszumokkal készültek. Ezen felül a switch-ek csomag-csonkolási és időbélyegző funkcióival is komoly gondok voltak. A megfizethető kategóriában

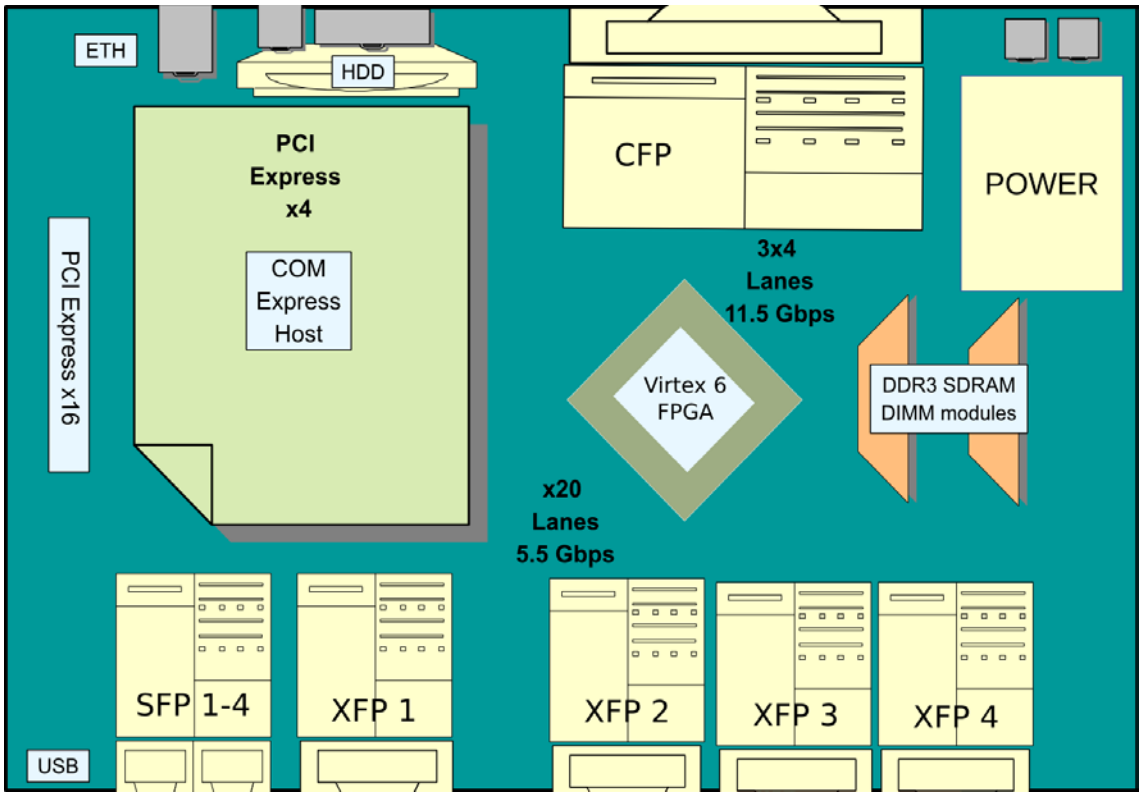
nem létezett olyan switch, ami ezt a három funkciót egyszerre kielégítő szinten tudja kezelni. Az IDS-em kiszolgálására egy új Intel chipset-tel rendelkező switch-et használtam. Az adatfeldolgozás kapacitásának fejlődése is csak elmúlt években tette lehetővé ilyen mennyiségű adat feldolgozását. Most jött el az idő, hogy a kör bezáruljon. A switch-ek eljutottak arra a szintre, hogy megfelelő minőségben tudják adat szolgáltatni. A cél-hardware alapú adatfeldolgozás pedig arra a szintre jutott, hogy valósidőben reagáljon.

5.3. A rendszer funkcionalitása

A rendszer nem csak telepítésében egyszerű, hanem könnyen konfigurálható is. A rendszer alkalmas SDN döntés hozás támogatására, a nagy felbontású, de könnyen feldolgozható információival.

5.4. C-GEP

A C-GEP[41][22] az AITIA Zrt. által gyártott, FPGA alapú 100 Gigabit/s Ethernet-képes csomagfeldolgozó platform (lásd 9. ábra). A C-GEP egy Virtex 6-os Xilinx FPGA-t használ a csomagfeldolgozáshoz. A nagy erejű újra konfigurálható hardware-nek köszönhetően képesek is vagyunk feldolgozni a forgalmat, ami érkezik a 100G-s linken [11]. Az FPGA-ra implementált cél-hardware teljesítményével csak nagyon drága és komplex felhő alapú rendszerek tudnak versenyezni. Viszont az FPGA 315 MHz-en üzemmel 100Gbit/s-es konfiguráción, ez a magas frekvencia lényegesen megnehezíti és lelassítja a tervezést. Az FPGA erőforrásai sokkal korlátoltabbak, mint egy PC alapú rendszernek, valamint sokkal nehezebben bővíthetők, de sokkal effektívebben felhasználhatóak. Ezért célszerű más elemeket is bevonni a rendszerbe.



9. ábra C-GEP blokk diagram [22]

5.5. 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. A kommunikációs protokollnál IPv4/UDP-ra esett a választás, tehát UDP datagram 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.

5.6. Az IDS software

Az IDS software komponense nincs konkrét termékre szűkítve; többféle megoldás is használható. Fontos követelmény vele szemben, hogy nyílt forráskódú legyen és könnyen módosítható, hiszen a C-GEP API-n átadott jelzéseket értelmeznie kell tudni.

5.7. A switch

Nagyon fontos szempont volt, hogy erős mirror funkcióval legyen ellátva. A rendszer képtelen lenne működni egy jó mirror nélkül. Tudnia kell csonkolni (truncating) a mirroron kiadott csomagokat, és timestamppel ellátni. A switch elterjedtsége is fontos szempont volt az adapter switch választásánál.

5.7.1. A mirror

A mirror eredeti célja az volt, hogy a mienkhez hasonló monitorozó rendszereket lásson el adattal. Ez nem egy új technológia, viszont kommersz switcheknél nem volt prioritás a hatékony megvalósítása.

5.7.2. 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 sávszélesség 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.

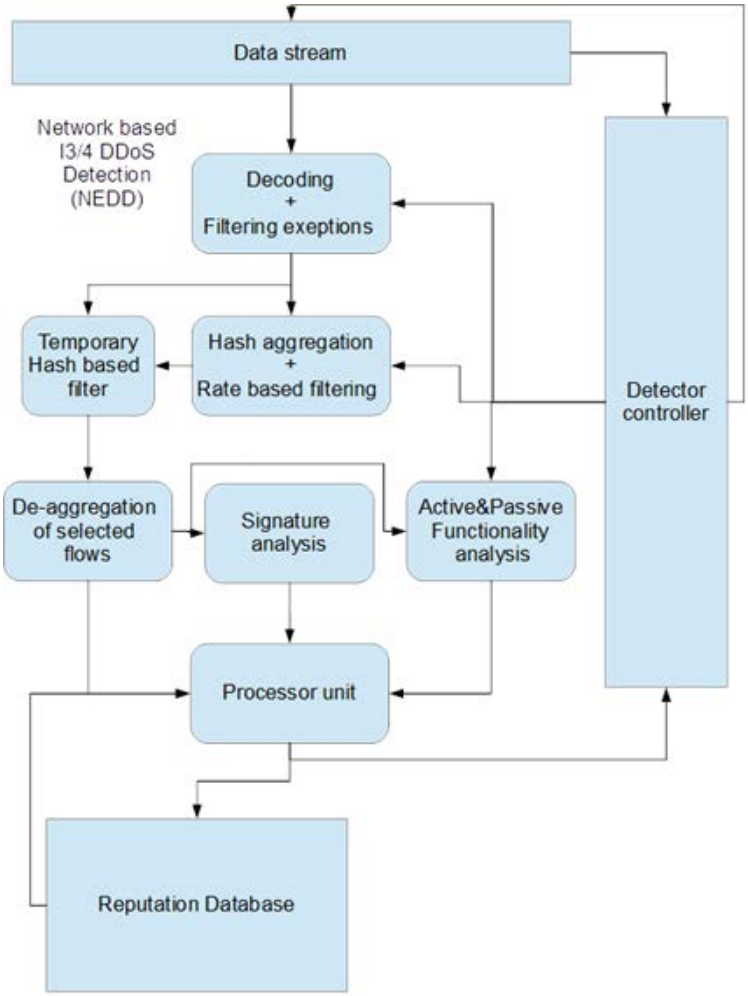
5.7.3. Az időbélyegzés

A timestamp (időbélyeg) talán a legkevésbé magától értetődő szükséges funkció a switchen belül. Ha a rendszert egy nagyobb hálózatra akarjuk installálni, nem csak egy ponton kell mintavételeznünk a hálózatból, nem csak egy C-GEP-pel. Ahhoz, hogy koherens képet kapjunk a hálózatról, a feldolgozandó csomagokat nem csak térben (MAC), de időben is pontosan el kell tudni helyezni. Szükséges tehát eltárolni, hogy adott csomag mikor „volt” a switchen ahol mintavételeztük. Ideális lenne, 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 mirror-ján nem feltétlenül sorrendhelyesen adja ki a csomagokat, maga a mirror-ozás rakna egy kisebb bizonytalanságot az időbélyegzésbe.

5.8. A SDN controller

Jelenleg a rendszer nem működik együtt SDN rendszerrel. Ha tartalmaz SDN controllert a hálózat, akkor kiválóan alkalmas annak a döntéshozatalát segíteni. A switchek (különösen az olcsó ToR switchek) limitált segítséget tudnak nyújtani a SDN döntéshozásban, mivel ezen eszközök 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 controllernek a döntéshozatalhoz minél részletesebb információra van szüksége, ebben a C-GEP adatfeldolgozó platform segítséget nyújthat. A SDN koncepció a támadások effektív kiszűrésében hatalmas segítséget nyújthat. A SDN-nek köszönhetően a (D)DoS támadás csomagjait a támadóhoz sokkal közelebb tudjuk kiszűrni.

6. FPGA alapú DDoS detektor tervezése (NEDD)



10. ábra A detektor elvi működése

A legtöbb DDoS támadás detektálásához flow vagy legalább végpont szintű megfigyelés szükséges. Nagy hálózatokban nagyon sok gyors elérésű memóriát igényelne minden flow megfigyelése. Ezért nem minden flow-t figyelünk, egyszerre hanem egy több lépcsős megoldást választunk a design alapjául. A feladat célja volt, hogy a DDoS támadásokat kis válasz idővel tudja detektálni a rendszer, ez a kritérium nagyon fontos szerepet játszott a tervezés során.

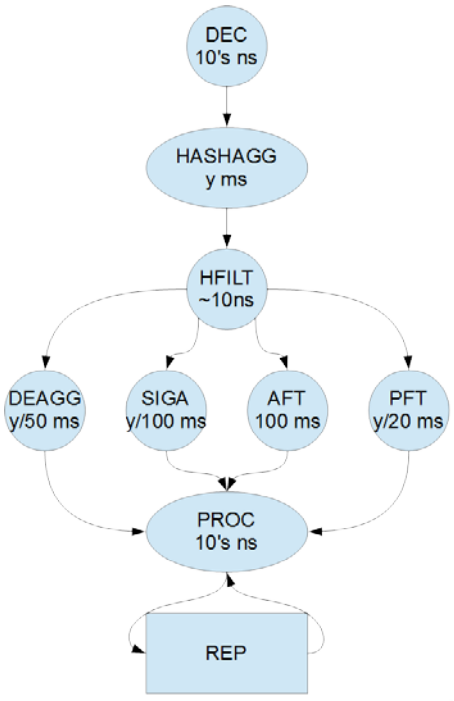
A támadás detektálása három fő lépésből áll. Egy dekódolási lépésből, egy BRAM memória alapú cél IP-re osztályozott csomag-sebesség mérésből, és egy végpont alapú támadás vizsgálatból.

A 10. ábra az egyik első terv, ami a funkciók szerinti tervezés alapján jött létre, tehát nem a valós modulokat mutatja be, hanem a kívánt funkciókat. A dekóder ezt a VHDL-ben megírt részét NEDD-nek (FPGA based **Ne**twork level **DD**oS detector).

6.1. Többlépcsős dekóder koncepció

A koncepció alapvető célja az, hogy a DDoS támadás detekció ne egy lépésben történjen meg, mivel egy nagy bizonyosságú döntéshez sok, nagy erőforrás igényű vizsgálatot kell elvégezni. Ezért érdemes a vizsgálatokat sorosítani, lebontani több lépésre. Minden egyes lépés végén érdemes eldobni azokat az end-pointokat amik ellen kizárható, hogy támadva lennének. Ezzel a metódussal bár valamelyest csökken a feldolgozási sebesség, nagyon sok erőforrást lehet megtakarítani.

6.2. A detektor válaszideje



11. ábra A detektor válaszidejének vázlata

A 11. ábra mutatja a detektor válaszidejének egy hozzávetőleges vázlatát; saját méréseken alapuló becslésekből kiindulva.

Látható, hogy a „HASHAGG” (hash készítés és osztályozás) modul és a „AFT” (aktív funkcionalitás lekérdezés) a leglassabbak – a többi modul legalább egy nagyságrenddel gyorsabb. Az „AFT”-nél ez annyira nem kritikus, mivel az egy extra funkció, kis konfidenciájú döntések segítéséhez. Viszont a „HASHAGG” minden folyamat alapja, ezért nagyon hasznos lenne gyorsítani a működését. Erre lehetőséget ad az úgynevezett szekvenciális analízis [23], ami minden egyes esemény után kiértékel, szemben a hagyományos analízissel, ami csak a periódus végén értékeli ki az eredményeket.

Az y értékét nehéz volt megmondani az implementáció előtt, de tudtuk, hogy a 0.1ms-10ms időtartományban kell tartani ahhoz, hogy teljesítse a követelményeket.

6.3. A dekódolás tervezése

A dekódolás minden csomag-feldolgozási ciklus első lépése. A dekódolás lényege az, hogy a bonyolultan beágyazott csomagokból kinyerjük a számunkra lényeges információt.

A dekóder modulnak el kell bírnia minél több fajta bonyolult protokoll-egymásba-ágyazással (enkapszulációval).

A dekódernek a C-GEP belső (SPI-hoz hasonló) buszából kell kinyernie az adatokat, ez több fajta konfigurációban is történhet. A dekódernek tudnia kell működnie több fajta sebességen is. A 1-10Gbps 100MHz körüli FPGA órajel-frekvenciákat igényel, viszont a 100Gbps 312MHz-s FPGA órajel-frekvenciát igényel.

6.3.1. A dekóder által dekódolt mezők

A dekóder a döntéshez szükséges mezőket kell, hogy dekódolja. Ezek a következők:

IPv4 header: Source IP, Destination IP, Protocol, Total length, Flags, Fragment offset, TTL

UDP header: Source port, Destination port

TCP header: Source port, Destination port, Flags

ICMP header: Type

6.3.2. Data checksum

A legtöbb direkt, volumetrikus támadás ugyanazt a klónozót csomagot küldi el nagyon sokszor, ezért célszerű a data részből is valamit kimenteni a feldolgozó számára. Mi egy checksum-os megoldást választottunk erre. Eszerint az érkezett adat (data) első „n” byteját összeadja a dekóder és továbbítja a feldolgozónak.

6.3.3. Kompatibilitás

Felmerül az igény, hogy a detektor ne csak 100Gbit/s interface-szel tudjon dolgozni, hanem lassabb interface-ekkel is, ezért kell tervezni kompatibilitási modulokat 1-10Gbit/s interface-ekhez is.

6.4. Az end-point terheltség mérés

Mivel volumetrikus támadásokra fókuszálunk, ezért érdemes kiszűrni azokat a végpontokat, amik egy bizonyos csomag-sebesség alatt kapnak adatot: e végpontok ellen nem folyik a specifikáltnál nagyobb támadás.

A specifikált sebesség-határt célszerű úgy megállapítani, hogy on-the-fly konfigurálható legyen.

6.5. DDoS detektor modul

Ez a modul hagyományos szignatúra és anomália analízist hajt végre. Ez az előbbi modulokkal ellentétben nem egy szolgáltatást, hanem többet valósít meg, párhuzamosan futatva őket. A modul tervezésénél cél volt az Akamai 2016 Q4-es [3] security report-jában szereplő kilenc leggyakoribb támadás detektálása. Ezen felül minél több volumetrikus támadás detektálása.

A DDoS detektor a különböző algoritmusokat tartalmazza:

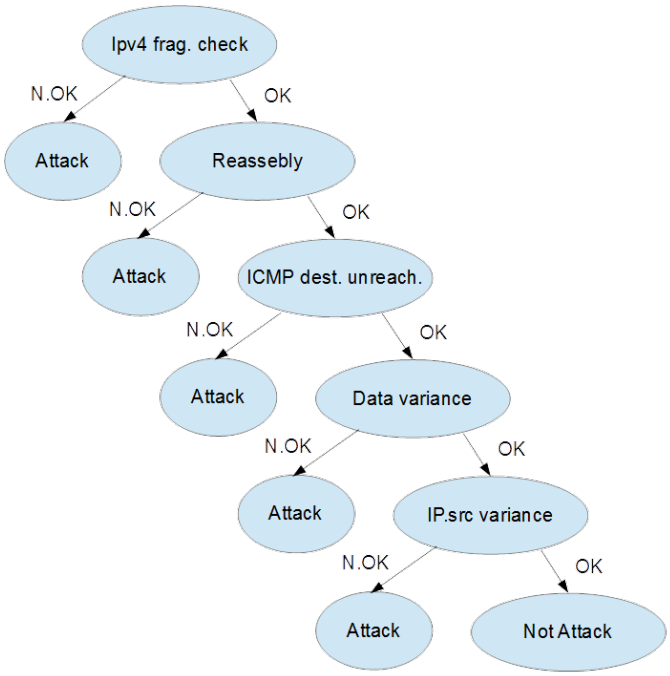
- **NTP, DNS, CHARGEN, RiPv1, SSDP visszavert támadás detekció**: Ez nem egy algoritmus, hanem több nagyon hasonló algoritmus, ami azon a definíción nyugszik, hogy a visszavert csomagokat nem kérte az áldozat. Tehát a lekérdezések és a válaszok arányát vizsgáljuk.
- **UDP fragmentation gyönge algoritmus**: Ennek az algoritmusnak a lényege, hogy az első fragmentált csomagok (MF=1 Offset=0) és utolsó fragmentált csomagok (MF=0 Offset!=0) arányát méri. Ez az algoritmus alkalmas az egyszerűbb (és egyben leghatásosabb) fragmentált támadások kiszűrésére ahol csak egy fragment kezdő csomagot küldenek.
- **UDP fragmentation erős algoritmus (avagy de-fragmentáció)**: Ez az algoritmus megpróbálja helyre állítani a fragmentált csomagokat bizonyos timeout feltételek mellett ez minden hagyományos fragmentation támadást képes kiszűrni.
- **SYN flood algoritmus**: Ez az algoritmus a SYN flood támadások kiszűrésére szolgál. Az algoritmus a TCP protokoll különböző jelző bitjeit figyeli az adott flowokban. Figyeli a SYN, ACK, SYNACK, DATA üzenetek arányát, ezzel viszonylag nagy konfidencia szinttel kiszűrhetjük a SYN floodokat.
- **SYN flood spoof algoritmus**: Ez az algoritmus a spoofolt SYN floodok kiszűrésére szolgál. Arra a mechanikára épít, hogy amikor az áldozat egy spoofolt SYN üzenetre küld egy SYN-ACK üzenetet amire RST-ACK üzenetet kap válaszul mivel az, aki megkapja a SYN-ACK üzenetet, ö nem küldött SYN üzenetet.
- **ICMP flood algoritmus**: Ez az algoritmus „csak” a bejövő ICMP csomagok sebességét számolja.
- **Source IP variancia**: A DDoS támadások két nagy csoportját (DoS és spoofolt DDoS) a source IP-k varianciája nagyban megkülönbözteti a valós forgalmaktól. Ez nem egy bizonyos típusra triggerel, hanem bizonyos megvalósításokra.
- **Passzív áldozat analízis**: Ez a modul a vélt áldozat hálózati aktivitását méri, ezzel segítve a döntéshozást.
- **Ezen felül 7 algoritmus a direkt UDP támadásokra, amit a következő alfejezetben ismertetek.**

6.6. UDP direkt támadások döntési fája

Az UPD (fragmentation flood, flood) alapú támadások nagy biztonságú detektálása a legnehezebb feladat, mivel ezek a támadások a legáltalánosabbak. Sajnos ezekre a támadásokra nincs egy egységes detekciós algoritmus, ezért több különböző gyengébb algoritmus összegzése alapján dönt ezekről az eseményekről, hogy támadások-e.

UDP algoritmusok:

- **IPv4 fragmentation statisztika**: Ennek az algoritmusnak a lényege, hogy az IPv4 fejlécben megtalálható „flags+fragmentation offset” mezőkből, minden csomaghoz dekódol két változót. Az egyik változó a „First fragment” ami egy, ha a „More fragment” = 1 és az „offset” nulla. A másik változó „Last fragment” ami egyes értékű, ha a „MF” = 1 és az „offset” != 0 értékű. A statisztika lényege, hogy ha az adott end-point-hoz tartozó „FF”-k száma x-szel nagyobb, mint a „LF”-k száma akkor biztos, hogy támadásról beszélhetünk.
- **IPv4 reassembly**: Ez az algoritmus a kifinomultabb fragmentation támadásokra fókuszál. A lényege az, hogy megnézze, hogy a beérkező töredezet csomagokat össze-e lehet-e állítani; ehhez a tárolja minden flow „identification”, „fragment offset” mezőit.
- **ICMP destination unreachable statisztika**: Az UDP floodok egyik leggyakoribb eleme, hogy random portokra küldik a csomagokat a hatás maximalizálás érdekében. Az áldozat ezekre a csomagokra ICMP destination unreachable üzenetekkel válaszol, amiket a detektor számol, és ha azt látja, hogy ezeknek az üzeneteknek a száma nagy, akkor jelzi, hogy támadás van folyamatban.
- **Adat checksum szórásnégyzet**: A legtöbb direkt UDP támadás ugyanazt az üzenetet klónozza nagyon sokszor. Ezért van egy olyan detekciós eljárás is, ami a csomag adat mezőjéből készít egy checksum-ot. Az adott end-point-ba beérkező csomagok adat checksum-jából számít szórásnégyzetet, ha nulla közeli ez az érték, akkor kijelenthetjük: ez egy klónozott üzenet.
- **IP source szórásnégyzet**: Ez a módszer elsősorban a spoofolt támadások kiszűrésére szolgál ahol a source IP címek általában random kerülnek a csomagba. A módszer lényege hogy az adott end-point-ra beérkező source IP címekből egy XOR hash-t készít, majd azokból szórást számít. Ha a szórás értéke nagyon nagy, akkor valószínűsíthető, hogy spoofolt támadásról van szó.



12. ábra Az UDP alapú támadások döntési fája

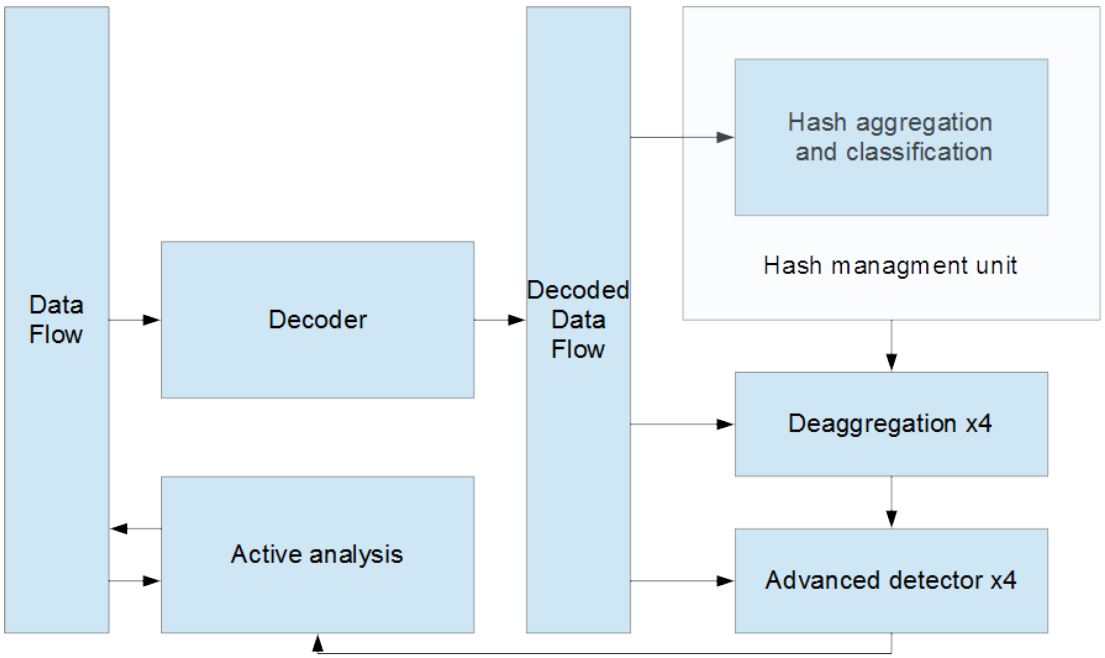
A 12. ábra ezen algoritmusok precedanciáját mutatja be. Valójában ezek a tesztek egyszerre hajtódnak végre, és a kiértékelésük is párhuzamos, és a tesztek eredményei között logikai vagy kapcsolat található.

6.7. Reputáció alapú adatbázis

Egy reputáció alapú adatbázis minden komolyabb detektor rendszer fontos eleme, ezért én is beleterveztem az eredeti design-ba. A reputáció adatbázist FPGA-ra implementálni sajnos kifejezetten nehéz és értelmetlen, mivel egy ilyen adatbázisnak nagyon kicsi a számítási igénye, viszont nagyon nagy a memória igénye. Utóbbival az FPGA csak szűkösen gazdálkodhat. Mivel a dolgozatom témája FPGA alapú tervezés, ezért ennek keretében nem fogok ilyen adatbázist elkészíteni. Egy ilyen adatbázissal a kommunikáció külön standard hálózati interface-en történhet.

7. FPGA alapú DDoS detektor implementálása

Egy hardware alapú teljesítmény-orientált eszköznél az implementáció lehetőségei nagyban befolyásolják a tervezés lépéseit ezért több tervezési részletet ebben a fejezetben fogok tárgyalni. Az implementáció a Xilinx ISE [20] segítségével történt.



13. ábra A detektor felépítése

A 13. ábra mutatja a támadás detektor moduláris felépítését, ez az ábra már valódi fizikai részeket tárgyal.

A detektor moduljait célszerűbb ilyen lebontásban tárgyalni.

7.1. A magas órajelű tervezési megkötések

A 100Gbps-s interface használatához 312 MHz órajel frekvencia tartozik, ami komoly hardware/firmware-tervezési megkötésekkel jár.

Minden modul kimeneteit és bemeneteit pufferelni kell. Azért, hogy az időzítési megkötésekbe beleférjen, nem lehet olyan logikákat használni, ami több logikai blokk soros összekötését igényelné egy órajel alatt. Az egymás mellé köthető BRAM-k száma korlátos, szintén az időzítési megkötések miatt.

7.2. Dekóder

A dekódert ebben a fejezetben csak röviden tárgyalom. A detektor jelenlegi állása szerint Ethernet, IPv4, TCP, UDP, MPLS, L2TP, PPP header-eket tartalmazó csomagokat tud dekódolni. Azért ezekre a header-ekre esett a választás, mivel az általam vizsgált hálózatban ezek a csomagok teszik ki a teljes forgalom 90%-át. A dekóderhez készült 1Gbit-es és 10Gbit-es kompatibilitási modul is.

7.3. 1Gbit/s, 10Gbit/s kompatibilitási modul

A dekóder 100Gbit/s-os interface-re lett tervezve, viszont a teszteléshez és bizonyos applikációkhoz ez az interface nem ideális, ezért ehhez illeszteni kellett a detektor több modulját. A dekóder az egyik ilyen modul. Ahelyett, hogy több dekódert készítettünk volna, azt a megoldást választottuk, hogy a dekóder és az interface közé kompatibilitási modulokat illesztetünk be.

A kompatibilitási modul a 1-10G-s interface-k felhasználásával szimulálja a 100Gbps-s interface-t.

7.4. Hash aggregáció és osztályozás

Ez a modul két részből áll, az IP-cím hash-t készítő algoritmusból és az osztályozó, sebességmérő algoritmusból. Ennek a két algoritmusnak célszerű effektíven együtt működnie. A hash-elés minősége fogja megszabni a sebességmérés pontosságát.

7.4.1. A hash algoritmus

A hash algoritmussal szemben két fő követelmény van: egyrészt az algoritmusnak 32 bit 16 bit átalakítást kell végre hajtania, másrészt a hálózati IP címeket egyenletesen kell elosztania a hash-ek közt. Ezen felül könnyen portolhatónak kell lennie a különféle sebességű, FPGA-alapú hardware-ekre.

Két különböző algoritmusra esett a választás tervezés során.

1) Xor hash: Hash (15 downto 0) <= IP (31 downto 16) xor IP (15 downto 0)

Ennek az algoritmusnak a nyilvánvaló előnye a könnyű implementálhatóság, viszonylagosan jó hatásfok, egy órajeles végre hajthatóság bármilyen órajel frekvencia mellett.

2) Jenkins hash [24]:

A Jenkins hash-re azért esett a választás, mert ez egy jól dokumentált, bizonyítottan jó algoritmus. Viszont ezzel az algoritmussal az a baj, hogy Neumann architektúrára lett fejlesztve, azaz szekvenciális feldolgozást feltételez, ami FPGA-nál nem biztos, hogy bölcs döntés. Ez sajnos igaz a legtöbb elterjedt hash algoritmusra is. Rövid tesztelés után beláttam, hogy a Jenkins hash szekvenciális formájában nem implementálható FPGA-ra komoly teljesítmény csökkenés nélkül. Innen két irányba lehetet elindulni:

1. Jenkins hash párhuzamosítása matematikai eszközökkel
2. Jenkins hash alapján készült FPGA-ra szabott hashelési algoritmus

Az első megoldás előnye, hogy a Jenkins hash jó tulajdonságait megőrzi, és annak a dokumentáltságát is, azaz bizonyítottan jó lesz. Viszont az algoritmus sajátosságai miatt nem minden szekvenciális rész lesz párhuzamosítható, valamint nagy órajel frekvencián bizonyos aritmetikai műveletek miatt további pipeline művelet végzőket kell beiktatni az időzítési feltételek betartásához.

A második megoldás előnye, hogy maximálisan párhuzamosítható: akár két órajel alatt is végrehajtható, és megőrzi a Jenkins hash jó tulajdonságainak legalább egy részét. Viszont teljesen teszteletlen, bizonyítatlan ez a gyakorlatilag új algoritmus.

Hosszú mérlegelés után a választás a második opcióra esett, mivel az első opció megvalósítása 315 MHz órajel mellett nagyon komoly gondot okozott volna.

7.4.2. A sebesség mérés

A sebességmérés nagyon fontos lépés a támadások észlelésében. Ezzel a lépéssel tudjuk kiszűrni az alacsony sebességű flow-kat a vizsgált flow-k halmazából.

A sebességmérő tervezésénél három fontos szempontot vettem figyelembe. A legfontosabb szempont a megvalósíthatóság. A második legfontosabb a párhuzamosan nyilván tartható flow vagy end-point sebességek száma. A harmadik a használandó memória típusa: mivel nagy mennyiségű sebesség-információt akarunk számon tartani, és a csomagok, amik a sebességet befolyásolják szekvenciálisan érkeznek be, ezért Blokk RAM alapú megoldásra esett a választás. A blokk RAM előnye, hogy sokkal több elérhető belőle, mint a slice memóriából, viszont lassabb az elérése. Az egyesített blokk RAM méret kompromisszumokkal, 20*36kbit lett. Minél nagyobb az egyesített tábla, annál több adat fér el benne – viszont a méret növekedésével csökken a tábla teljesítménye, minél több blokk RAM-ot kötünk össze, annál több buffer fokozatot kell beiktatni. A jelenlegi 20 blokk RAM-os táblával 315 MHz-en 4 órajel hosszú olvasás művelet, két órajel hosszú írás, 16 bites cím busz és 10 bites adat cellák valósíthatóak meg. Érdekesség kedvért a 40 blokk RAM-os design 315 MHz-n: 6 órajel hosszú olvasás művelet, két órajel hosszú írás, 17 bites cím busz és 10 bites adat.

A sebesség méréshez szekvenciális analízis a kézenfekvő választás, mivel blokk RAM-ból készül.

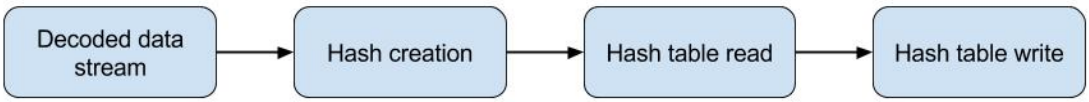
Érdemes ebben a lépésben specifikálni, milyen érték lesz a küszöb érték. Valamint érdemes bitsebesség helyet csomag sebességre triggerelni, mivel a csomag sebesség sokkal jobb metrika egy támadás erősségére, mint a bitsebesség.

Valós támadás sebesség Gb,	Csomag sebesség Csomag/s (kis csomagok)	Minimális mintavételezési idő ms	Csomag sebesség Csomag/s (nagy csomagok)	Minimális mintavételezési idő ms
0,1	208333,33	4,80	8333,33	120,00
0,5	1041666,67	0,96	41666,67	24,00
1	2083333,33	0,48	83333,33	12,00
2	4166666,67	0,24	166666,67	6,00
5	10416666,67	0,10	416666,67	2,40
10	20833333,33	0,05	833333,33	1,20
20	41666666,67	0,02	1666666,67	0,60
50	104166666,67	0,01	4166666,67	0,24
100	208333333,33	0,00	8333333,33	0,12
200	416666666,67	0,00	16666666,67	0,06
500	1041666666,67	0,00	41666666,67	0,02
1000	2083333333,33	0,00	83333333,33	0,01

14. ábra Táblázat a mintavételezési időkről

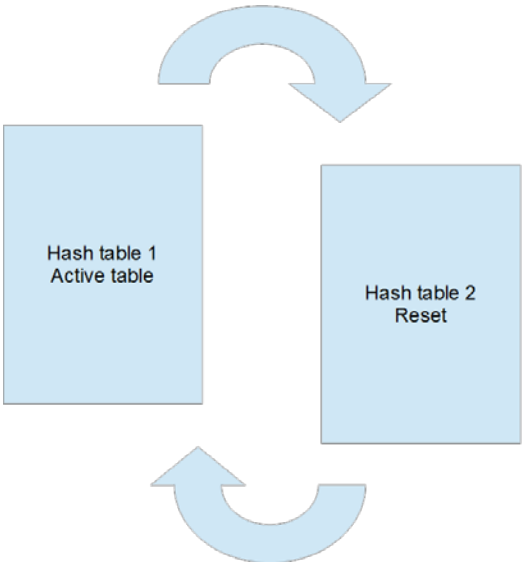
A 14. ábra mutatja a minimális mintavételi periódus időt különböző támadási sebességekhez, ha azt feltételezzük, 1000 azonos végpontú csomag elég erős konfidencia szintet biztosít. Látható, hogy a kitűzött 10ms felsőhatárt csak a legextrémebb esetekben lépi túl a trigger idő. Ezekben az esetekben is gyorsabb trigger időkre lehet számítani, mint a táblázatban szereplő értékek, mivel worst-case csomag eloszlásokkal számoltam.

7.4.3. Az osztályozó architektúrája



15. ábra Az osztályozás lépései

A 15. ábra mutatja be az osztályozó alapvető logikáját. Először létrehoz egy hash-t a destination IP-ből, majd kiolvassa a hash táblából, hogy az adott időszakban hány megegyező hash-ű csomag érkezett, azt eggyel inkrementálja, ezt az értéket majd visszaírja a táblába. A táblát on-the-fly konfigurálható időnként reseteljük.



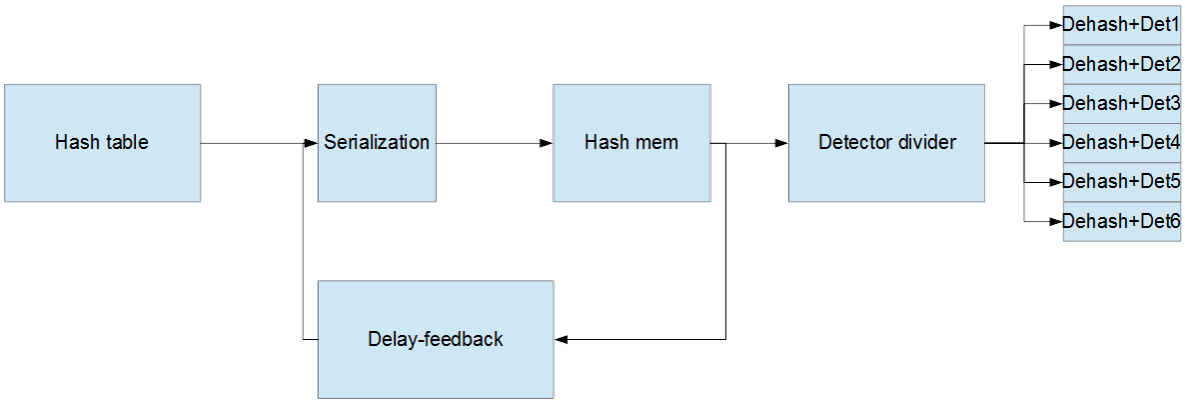
16. ábra A resetelés logikája

Mivel a Xilinx BRAM-okba szigorúan egyszerre csak egy cella változhat, ezért a reseteléshez még egy hash táblát be kell vonni a folyamatba 16. ábra. Amíg az egyik táblán aktív munka folyik, addig a másik tábla tud „resetelődni”, nullákat írunk minden cellájába.

Mivel a sebességmérő szekvenciális analízisen alapul, ezért minden bejövő csomagnál ellenőrzi, hogy a sebesség-kritérium teljesül-e.

7.4.4. Osztályozás management

A hash tábla megvalósításának hátránya, hogy ugyanaz a hash, ha nagy mennyiségben ékezik, újra és újra triggerelheti. Ezért egy extra management modult kellett elhelyezni a folyamat végére.



17. ábra Az osztályozás management működési elve

Az 17. ábra szemlélteti a management működését. A hash mem tárolja milyen hash-k triggerelték a hash tábla kimenetét. Ezt úgy érjük el, hogy amikor egy hash-t kiad a hash tábla, az bekerül a hash mem-be – ha eddig nem volt benne –, ezzel párhuzamosan belekerül egy késleltetési modulba, ami 1 ms-al később kidob egy parancsot, ami tartalmazza azt a hash-t, majd ez a parancs törli a hash-t a hash mem-ből.

7.5. De-aggregáció

A de-aggregáció az a lépés, amikor a kiválasztott trigger feltétel feletti hash-ekből kikódoljuk a támadó végpontok IP címeit. Mivel a hash készítés definició szerint információvesztéssel jár, ezért további méréseket kell végezni az IP címek kitömörítése érdekében.

Ez egy slice-ba implementált CAM memória segítségével történik meg. A CAM feladata, hogy detektálja azokat a címeket, amik túllépik a specifikált sebesség értéket. A CAM implementációról azt kell tudni, hogy nehéz jól implementálni, és ahogy nő a mérete, úgy csökken a sebessége. Szerencsére a hash szűrés egy normális hálózatban elég erős ahhoz, hogy 16 helyes CAM elég legyen erre a célra. Ebben a modulban a CAM architektúra miatt sima (nem szekvenciális analízis) analízist alkalmazunk.

7.6. Detektor modul

A detektor modul a legkomplexebb az összes modul közül, mivel sok különböző algoritmus megvalósítását is tartalmazza. Ez a modul kizárólag slice-ban van megírva, és hagyományos analízist alkalmaz. A fentebb ismertetett algoritmusokat valósítja meg. A modul hagyományos analízissel működik a sok párhuzamos vizsgálat miatt. Implementáció szempontjából nem túl érdekes a modul, egész slice logikába van írva néhány DSP használatával (ami a szórás számításhoz szükséges).

7.7. Aktív analízis modul

A detektorhoz nem csak passzív modulok tartoznak. Lehetőség van arra, hogy a detektor a vélt áldozatokat ICMP üzenetekkel letapogassa, kiderítve, hogy milyen állapotban van a szolgáltatás. A modul implementációja nagyon egyszerű: egy külön 1G-s MAC inter-

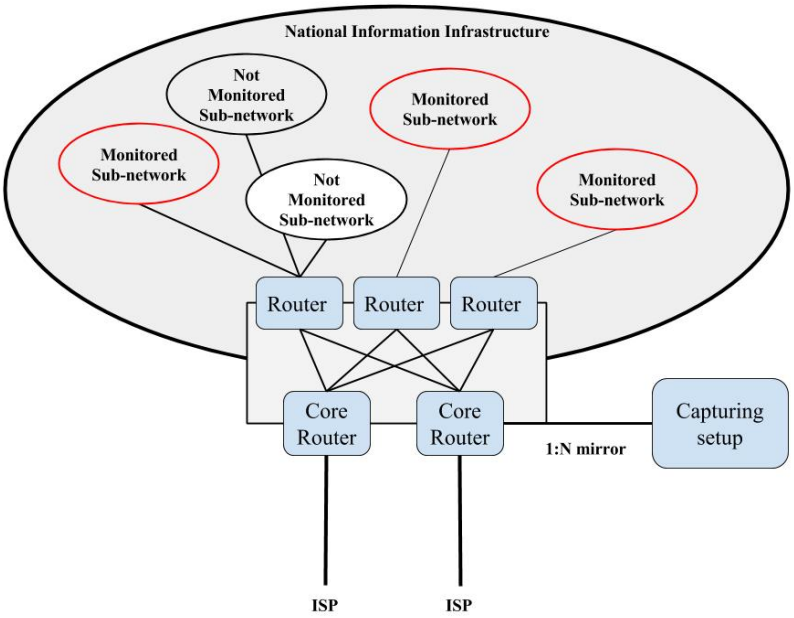
face-en ICMP lekérdezéseket küld ki, és azokra választ vár. A válaszidőkből átlagot számít, szórást számít és számolja az elveszett csomagokat, majd az eredményeket továbbadja a processzornak.

8. A detektor szimulálása

A detektor tesztelésének egyik legfontosabb lépése a szimuláció. A modul szintű szimuláció hagyományos tesztekkel készült – viszont az egység szintű szimuláció valódi, rögzített támadássokkal. Ehhez a teszt típushoz külön pcap olvasó test benchet készítettem, ami precízen kiolvassa a megadott fájlból az elmentett csomagokat. Az egyetlen probléma ezzel a test bench-el, hogy a dekóder modult nem tudja tesztelni.

8.1. A teszteléshez használt forgalom

A NIIFI (Nemzeti Információs Infrastruktúra Fejlesztési Intézet) rendszerében 2016-2017-ben gyűjtött forgalmat használtam a dekóder teszteléséhez. A NIIFI több alhálózatból áll, mint Sulinet, BME ezekből a hálózatokból kaptunk forgalmat egy 1:N-s mirroron 18. ábra. Ez a fajta teszt nagyon hasonlít a valós körülményekre, mivel minden egyes elemét valóságban rögzítettem.



18. ábra A NIIFI hálózat és a rögzítés blokk vázlata

8.2. A forgalom gyűjtés menete

A referencia forgalmat sima rögzítési (capture) módszerekkel készítettük egy nagy sebességű RAID meghajtóra.

A támadások mentésére egyszerű C++ -ban megírt kódot használtam, amiből kézzel válogattam ki a támadásokat, mivel az egyszerű C++ detektor rengeteg fals pozitív eredményt hozott.

8.3. A tesztelés menete

A teszt három részből áll: a támadás tesztből, a referencia tesztből, és a kevert tesztekből.

A támadás teszt az, amikor különböző támadások mintáit játszottam vissza a rendszernek és ezzel teszteltem a rendszer funkcionalitását.

Sikeresen teszteltem a rendszert NTP, DNS, UDP flood, UDP fragmentation, SYN flood, RIPv1 flood támadásokkal.

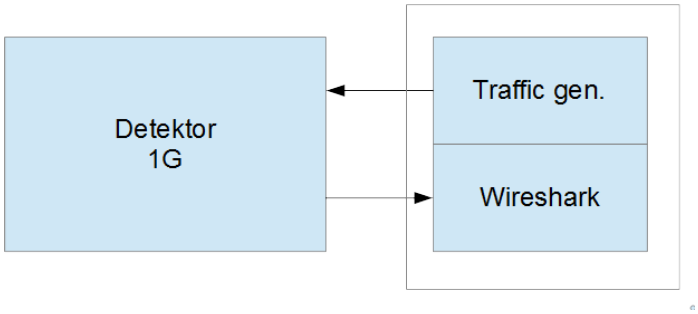
A referencia tesztnél normál, nem-támadás alapú forgalmat játszottam vissza. Ennél a tesztnél fals pozitív eredményeket kerestem. A tesztelés során nem találtam fals pozitív eredményt, viszont viszonylag kevés ilyen tesztet hajtottam végre ezért ez az eredmény kevésbé mérvadó, mint a támadás teszt.

A kevert teszteknél a támadás mintákat referencia mintákkal kevertem össze a szimuláció előtt ezzel tesztelve a rendszert. Ezek a tesztek sikeresen lefolytak semmi újdonságot nem hoztak.

9. A detektor tesztelése

A detektor validálásának egyik utolsó lépése a fizikai tesztelés. Ebben a fejezetben bemutatom, hogyan teszteltem a detektort.

9.1. A mérés összeállítása



19. ábra A fizikai tesztelés blokksémája

A fizikai teszteléshez a 19. ábra által szemléltetett összeállítást használtam. Ez az összeállítás egy PC-ből és egy 1G-s detektorból áll. A teszteléshez egy ipari PC-t használtam 1Gbps integrált hálózati kártyával. A forgalomgenerálást egy winpcap alapú házi készítésű pcap visszajátszóval valósítottam meg.

A teszt eredményeit a detektor az 1Gbps interface Tx kivezetésén vezetem ki és a Wireshark-kal elemzem.

A detektor kétfajta üzenetet küld: egy 3 másodpercenként ismétlődő diagnosztikai üzenetet, ami a detektor állapotait írja le, és a támadás észlelés üzentet, amit akkor küld, ha támadást érzékel a bemenetén.

9.2. A mérés menete

A mérés előkészítéséhez letiltom a hálózati kártya protokolljait, hogy csak az a forgalom kerüljön a detektor bemenetére, ami teszteléshez szükséges. Ezután elindítottam egy wireshark capture-t és elindítottam a forgalom lejátszását – ezt szemlélteti a 20. ábra.

20. ábra A forgalom generátor tipikus futtatási környezete

A teszteléshez ugyanazokat a forgalmakat használtam fel, amiket a szimuláció során használtam, csak most nem kiolvastam őket pcap-ből, hanem lejátszottam őket a sendcap-pel, ahogyan a 20. ábra szemlélteti.

9.3. A mérés eredményei

A mérés elsőre nem hozott sikeres eredményt, mivel az implementáció és a szimuláció közt komoly különbségek vannak. A diagnosztikai üzenetek segítségével debuggoltam a detektort. A debuggolási folyamat sikeres elvégzése után sikerült ugyanolyan eredményeket kapnom, amiket szimulációval előtte sikerült elérnem.

10. A támadás-tesztelés dokumentálása

Ebben a fejezetben a bemutatom a mindazokat a támadásokat, amiket a teszteléshez használtam, és az eredményeket, amiket nyertem a tesztelés során.

10.1. NTP támadás

Time	Source	Destination	Protocol
0.000035800	220.165.165.178	195.199	NTP
0.000036200	220.165.165.178	195.199	NTP
0.000036400	220.165.165.178	195.199	NTP
0.000036900	220.165.165.178	195.199	NTP
0.000040100	46.146.239.21	195.199	NTP
0.000070500	46.146.239.21	195.199	NTP
0.000076200	185.9.229.251	195.199	NTP
0.000086100	220.165.165.178	195.199	NTP
0.000089000	220.165.165.178	195.199	NTP
0.000094200	220.165.165.178	195.199	NTP
0.000101600	220.165.165.178	195.199	NTP
0.000101800	220.165.165.178	195.199	NTP
0.000102900	220.165.165.178	195.199	NTP

21. ábra Minta az NTP támadásból

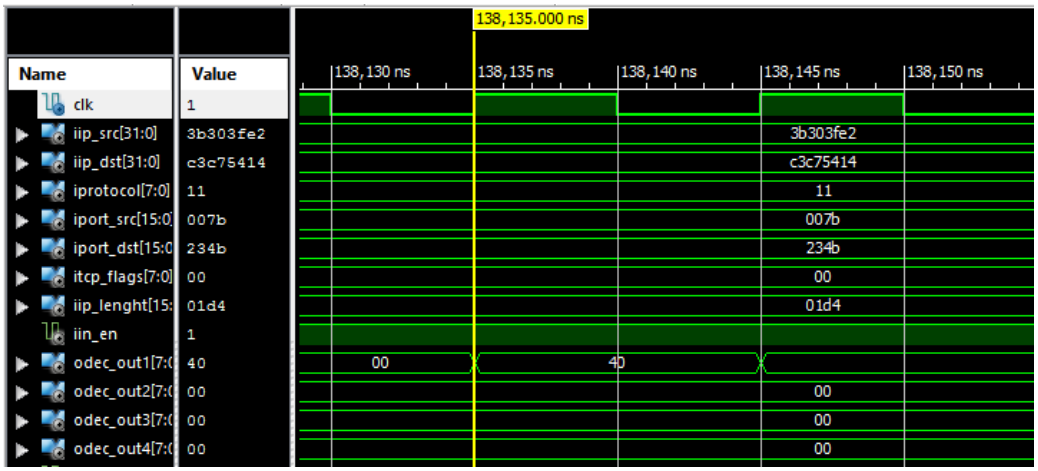
```
> Frame 2: 96 bytes on wire (768 bits), 96 bytes captured (768 bits)
> Ethernet II, Src: Cisco_0d:7a:6f (f8:66:f2:0d:7a:6f), Dst: Cisco_09:7d:13 (f8:66:f2:09:7d:13)
> MultiProtocol Label Switching Header, Label: 24154, Exp: 0, S: 1, TTL: 52
> Internet Protocol Version 4, Src: 220.165.165.178, Dst: 195.199
> User Datagram Protocol, Src Port: 123, Dst Port: 9035
v Network Time Protocol (NTP Version 2, private)
  > Flags: 0xd7, Response bit: Response, Version number: NTP Version 2, Mode: reserved for private use
  > Auth, sequence: 215
    Implementation: XNTPD (3)
    Request code: MON_GETLIST_1 (42)
    0000 .... = Err: No error (0x00)
    .... 0000  0000 0110 = Number of data items: 6
    0000 .... = Reserved: 0x00
    .... 0000  0100 1000 = Size of data item: 0x0048
  > Monlist item: address: 166.182.108.113:123
> [Malformed Packet: NTP]
```

22. ábra A támadás egy csomagja

Ez volt az első támadás, amivel teszteltem a rendszert. A támadás többszázezer NTP MON_GETLIST lekérdezés-választ tartalmaz, több különböző source IP-ről 21. ábra, 22. ábra.

10.1.1. Szimuláció eredményei

A szimuláció során a detektor helyes működést mutatott és jelezte, hogy NTP flood támadás van folyamatban. A támadást a rendszer 140us alatt észlelte és jelezte. Kijelenthetjük: ebben az esetben specifikációnak megfelelően működött a rendszer. A 23. ábra mutatja, hogy a szimuláció sikeres volt, az NTP támadás kódja 40h.



23. ábra NTP támadás szimuláció

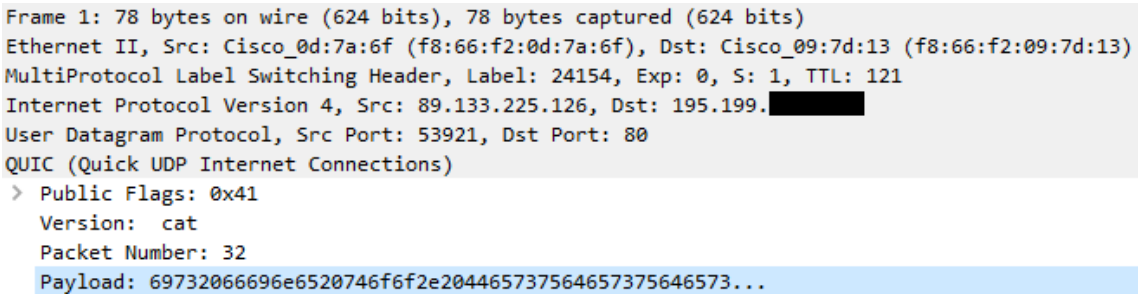
10.1.2. A tesztelés eredményei

A tesztelés sikerrel zárult: a detektor helyesen jelezte, hogy NTP támadás van folyamatban az áldozat ellen. Az eredményeket a specifikált időtartományban kaptam meg, pontos értéket itt nehezebb mondani mivel itt egyedül az alaplap által biztosított timestamp-elési funkció volt elérhető, ami egy komolyabb bizonytalanságot visz az időzítésbe.

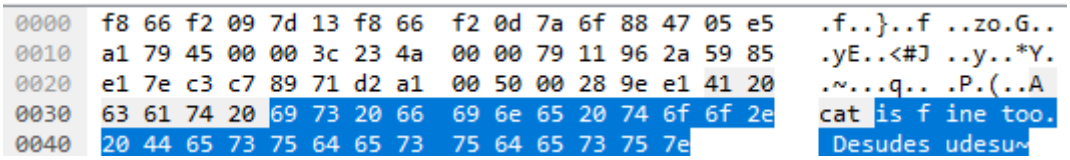
10.2. UDP flood

Time	Source	Destination	Protocol
0.000000000	89.133.225.126	195.199	QUIC
0.000003800	89.133.225.126	195.199	QUIC
0.000004000	89.133.225.126	195.199	QUIC
0.000004300	89.133.225.126	195.199	QUIC
0.000005300	89.133.225.126	195.199	QUIC
0.000007100	89.133.225.126	195.199	QUIC
0.000009200	89.133.225.126	195.199	QUIC
0.000011000	89.133.225.126	195.199	QUIC
0.000011200	89.133.225.126	195.199	QUIC
0.000011500	89.133.225.126	195.199	QUIC
0.000011700	89.133.225.126	195.199	QUIC
0.000014500	89.133.225.126	195.199	QUIC
0.000016700	89.133.225.126	195.199	QUIC

24. ábra A támadás néhány csomagja



25. ábra Az UDP flood egy csomagja



26. ábra Anonymus signaturája

A teszteléshez használt UDP flood (24. ábra, 25. ábra, 26. ábra) az ilyen támadások iskolapéldájának is felfogható, mivel ez a támadás az Anonymus DDoS eszközével [19] készült. Ez a payload-ból látható.

10.2.1. A szimuláció eredménye

A szimuláció során a detektor helyes működést mutatott és jelezte, hogy UDP flood támadás van folyamatban, nulla source IP szórással. A támadást a rendszer 200us alatt észlelte és jelezte. Kijelenthetjük, hogy ebben az esetben specifikációnak megfelelően működött a rendszer.

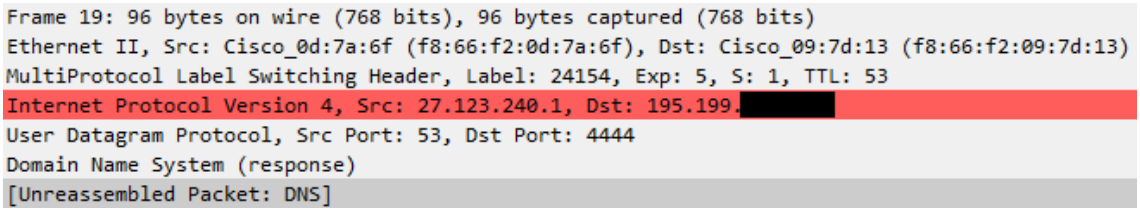
10.2.2. A tesztelés eredményei

A tesztelés sikerrel zárult. Nagyságrendileg 13000 elküldött támadó-csomag után megkaptuk a detektortól a támadást jelző üzenetet, amit helyesen készített el a detektor.

10.3. DNS flood

Time	Source	Destination	Protocol
22218.0832048…	85.115.224.18	195.199.[redacted]	IPv4
22218.0832049…	27.123.240.1	195.199.[redacted]	DNS
22218.0832085…	194.135.219.90	195.199.[redacted]	DNS
22218.0832096…	177.234.9.158	195.199.[redacted]	DNS
22218.0832113…	27.123.240.1	195.199.[redacted]	IPv4
22218.0832125…	177.234.9.158	195.199.[redacted]	IPv4
22218.0832126…	138.117.109.154	195.199.[redacted]	IPv4
22218.0832129…	79.160.90.234	195.199.[redacted]	IPv4
22218.0832132…	89.189.183.111	195.199.[redacted]	IPv4
22218.0832188…	185.120.191.27	195.199.[redacted]	IPv4
22218.0832217…	46.4.156.58	195.199.[redacted]	IPv4
22218.0832264…	27.123.240.1	195.199.[redacted]	IPv4
22218.0832346…	46.243.67.222	195.199.[redacted]	IPv4

27. ábra A DNS támadás néhány csomagja



28. ábra A DNS támadás egy csomagja

A teszteléshez használt támadás (27. ábra, 28. ábra) sok százezer DNS lekérdezés-választ tartalmaz, néhány ICMP üzenettel társítva.

10.3.1. A szimuláció eredménye

A szimuláció során a detektor helyes működést mutatott és jelezte, hogy DNS flood támadás van folyamatban, továbbá, hogy **NINCS** UDP fragmentation támadás folyamatban. A támadást a rendszer 140us alatt észlelte és jelezte. Kijelenthetjük, hogy ebben az esetben specifikációnak megfelelően működött a rendszer.

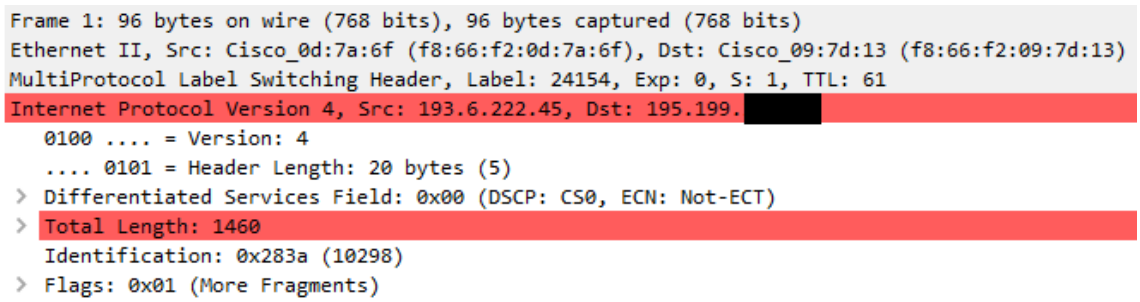
10.3.2. A tesztelés eredményei

A tesztelés sikeresen lezajlott a detektor DNS támadást jelzett.

10.4. UDP fragmentation attack

Time	Source	Destination	Protocol
0.000000000	193.6.222.45	195.199.	UDP
0.000008700	193.6.222.45	195.199.	UDP
0.000076700	193.6.222.45	195.199.	UDP
0.000092800	193.6.222.45	195.199.	UDP
0.000168700	193.6.222.45	195.199.	UDP
0.000186800	193.6.222.45	195.199.	UDP
0.000271500	193.6.222.45	195.199.	UDP
0.000283700	193.6.222.45	195.199.	UDP
0.000356900	193.6.222.45	195.199.	UDP
0.000370300	193.6.222.45	195.199.	UDP
0.000455200	193.6.222.45	195.199.	UDP
0.000464900	193.6.222.45	195.199.	UDP
0.000546000	193.6.222.45	195.199.	UDP

29. ábra UDP fragmentation támadás néhány csomagja



30. ábra Az UDP fragmentation támadás egy csomagja

A választott támadás (29. ábra, 30. ábra) sok százezer more fragment IPv4/UDP csomagot tartalmaz.

10.4.1. A szimuláció eredménye

A szimuláció során detektor helyes működést mutatott és jelezte, hogy UDP fragmentation támadás van folyamatban, és jelezte a csomagokat nem lehet helyreállítani. A támadást a rendszer 320us alatt észlelte és jelezte. Kijelenthetjük ebben az esetben specifikációnak megfelelően működött a rendszer.

10.4.2. A tesztelés eredményei

A tesztelés sikeresen lezajlott a detektor UDP fragmentation támadást jelzett.

10.5. RiPv1

Time	Source	Destination	Protocol
0.000000000	212.118.33.178	193.225.	RIPv1
0.000000200	206.168.40.84	193.225.	RIPv1
0.000000400	212.118.33.178	193.225.	RIPv1
0.000002000	206.168.40.84	193.225.	RIPv1
0.000006800	219.233.16.222	193.225.	RIPv1
0.000019300	210.134.118.36	193.225.	RIPv1
0.000020000	163.13.243.8	193.225.	RIPv1
0.000021800	89.239.16.238	193.225.	RIPv1
0.000023800	163.13.243.8	193.225.	RIPv1
0.000025600	89.239.16.238	193.225.	RIPv1
0.000034500	178.33.96.95	193.225.	RIPv1
0.000041400	134.91.35.6	193.225.	RIPv1
0.000052600	219.233.16.222	193.225.	RIPv1

31. ábra A RIP támadás néhány csomagja

```
Frame 5: 96 bytes on wire (768 bits), 96 bytes captured (768 bits)
Ethernet II, Src: Cisco_09:7d:13 (f8:66:f2:09:7d:13), Dst: Cisco_0d:7a:6f (f8:66:f2:0d:7a:6f)
Internet Protocol Version 4, Src: 195.111.97.250, Dst: 178.48.17.101
User Datagram Protocol, Src Port: 1701, Dst Port: 1701
Layer 2 Tunneling Protocol
Point-to-Point Protocol
Internet Protocol Version 4, Src: 219.233.16.222, Dst: 193.225.
User Datagram Protocol, Src Port: 520, Dst Port: 59256
Routing Information Protocol
[Malformed Packet: RIP]
> [Expert Info (Error/Malformed): Malformed Packet (Exception occurred)]
```

32. ábra A RiP támadás egy csomagja

A tesztelésre ezt a RiPv1 (31. ábra, 32. ábra) támadást választottam, mivel ez egy jellemző RiPv1 támadás. Több mint száz különböző Ripv1-s eszközt használt a támadó. Sajnos nem lehet tudni, hogy eredetileg hány forrást alkalmazott a támadás lefolytatásához.

10.5.1. A szimuláció eredménye

A szimuláció helyes eredménnyel zárult: a detektor 160us alatt jelezte a RIP támadást, ezzel sikeres volt a teszt.

10.5.2. A tesztelés eredményei

A tesztelés sikeresen lezajlott a detektor RIP támadást jelzett.

10.6. SYN flood

Time	Source	Destination	Protocol
0.000000000	219.231.48.99	46.101.	TCP
0.013400000	5.122.110.65	46.101.	TCP
0.013500000	66.131.25.62	46.101.	TCP
0.013700000	101.253.154.87	46.101.	TCP
0.014300000	216.247.246.106	46.101.	TCP
0.035800000	182.224.193.80	46.101.	TCP
0.042800000	183.51.195.27	46.101.	TCP
0.043600000	205.76.25.94	46.101.	TCP
0.047500000	159.1.116.82	46.101.	TCP
0.049600000	168.201.94.87	46.101.	TCP
0.049800000	61.236.105.112	46.101.	TCP
0.055900000	219.13.183.28	46.101.	TCP
0.057800000	108.136.146.37	46.101.	TCP
0.086800000	11.107.159.83	46.101.	TCP

33. ábra A SYN flood néhány csomagja

```
Frame 15: 62 bytes on wire (496 bits), 62 bytes captured (496 bits)
Ethernet II, Src: Cisco_09:7d:13 (f8:66:f2:09:7d:13), Dst: Cisco_0d:7a:6f (f8:66:f2:0d:7a:6f)
Internet Protocol Version 4, Src: 146.20.103.92, Dst: 46.101.
Transmission Control Protocol, Src Port: 6197, Dst Port: 21, Seq: 0, Len: 0
```

34. ábra A SYN flood egy csomagja

A detektor teszteléséhez ezt (33. ábra, 34. ábra) a SYN flood támadást választottam. Ez nem csak egy sima SYN flood, hanem egy spoofolt támadás több millió fiktív végponttal. Azért erre a támadásra esett a választásom mivel ez az egy direkt támadást rögzítettem spoof-olással.

10.6.1. A szimuláció eredménye

A detektor jelezte 210us alatt, hogy SYN flood támadás van folyamatban a protokoll statisztikák alapján. Emellett a detektor jelezte, hogy nagy valószínűséggel spoof-olás is van, mivel a source IP címek szórása hatalmas volt. Ennél nagyobb biztonsággal egyébként máshogyan sem nagyon lehet megállapítani, hogy spoof-olt a támadás, legalábbis jelenleg jól automatizálható módszerekkel nem.

10.6.2. A tesztelés eredményei

A tesztelés sikeresen megtörtént, a detektor ~15000 támadó csomag után helyesen jelezte, hogy SYN flood támadás van folyamatban, valószínűleg spoof-olt csomagok felhasználásával.

10.7. Szimulált eredmények kiértékelése

Ebben az alfejezetben összefoglalom az eredményeket és kiértékelem őket.

2. táblázat A szimulációs idők összehasonlítása a hat tesztelt támadásra

	NTP	**UDP**	**DNS**	**UDP FRAG**	**RIP**	**SYN**
DETEKCIOS IDŐK (UM)	140	200	140	320	160	210

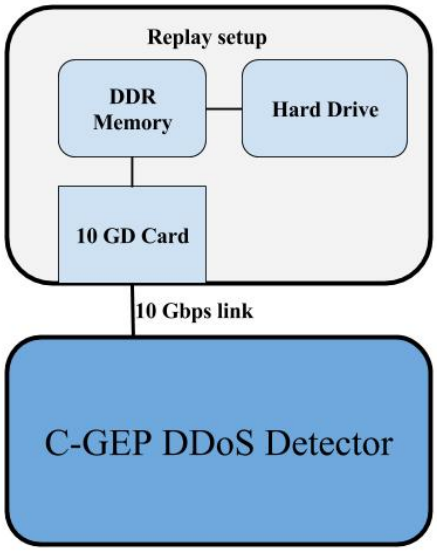
A 2. táblázat összefoglalja a szimulációs eredményeket. Látható, hogy minden egyes eredmény ezred másodperc alatt volt. Ezek nagyon biztató eredmények, de sajnos ebből még nem lehet messzemenő következtetéseket levonni. A szimuláció nem teszteli a dekódoló egységet, hanem a testbench annak a helyét veszi át. A szimulációban maximális teljesítményre való beállítással működött a NEDD. A szimuláció során 100Gbps-es testbenchet használtam, ami 10Gbps-n rögzített (capture) forgalmat felskálázta 100Gbps alapra, azaz 10-szer gyorsabban játszotta le.

Ezen okok miatt szükséges egy teljesen autentikus 10Gbps-s teszt sorozat, amit a következő fejezetben mutatok be.

11. A rendszer tesztelése 10Gbps Ethernet illesztéssel

Az 1Gbps tesztek elvégzése után lehetőség nyílt arra, hogy ezeket a teszteket 10Gbps berendezésekkel is elvégezzem. Ezeknek a teszteknek a végrehajtása módszertanilag megegyezett a korábbiakkal, a mérési környezet viszont más eszközökből állt. Az eredmények kiértékelése itt még az eddigieknél is részletesebben történt [39][38].

11.1. A tesztelési környezet



35. ábra 10Gbps tesztelési környezet

A tesztelési környezet két elemből áll a C-GEP-ből ami a detektort futatta és egy ipari PC-ből amiben a hálózati kártya helyett egy 10 GD FPGA kártya[22] volt installálva azért, hogy precízen vissza lehessen játszani a forgalmakat 10Gbps vonali rátán (35. ábra). Ez az összeállítás lehetővé teszi, hogy a pcap fájlokat nagy sebességgel és időbélyeg pontossággal visszajátsszuk.

11.2. Tesztelt támadások

3. táblázat – Behatolás-detekciós tesztelések 10Gbps Ethernet illesztéssel

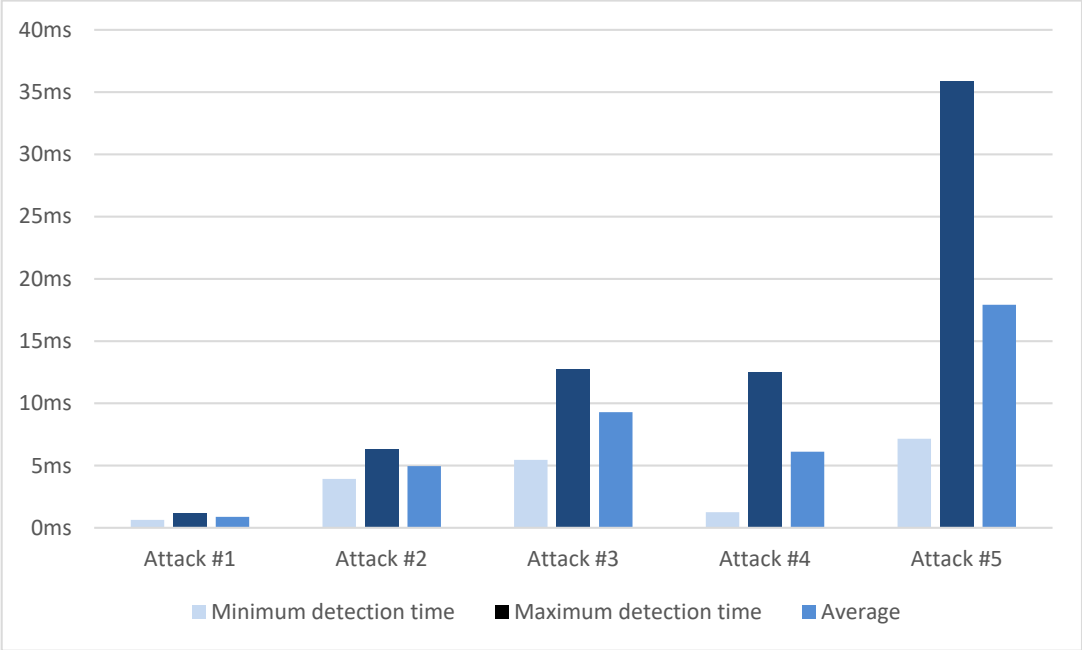
	Támadások				
	#1	#2	#3	#4	#5
Típus	DNS	SYN	NTP	QUIC	UDP fragmentation
Teszt hossza [s]	74	10	40	20	30

Intenzítás [Gbps]	7-8	0.5-0.6	1-2	0.23	0.15
Támadók száma	-	1	-	1	1
Reflektorok száma	~22000	-	~3500	-	-

A fenti 3. táblázatban szereplő támadásokra végeztem el a teszteket. A támadások forgalmát normális forgalommal keverve játszottam vissza. Ebben a tesztsorozatban teszteltem a pontos detekciós időket, valamint mértem a támadás hány csomagja jut át a rendszeren a támadás detektálásáig.

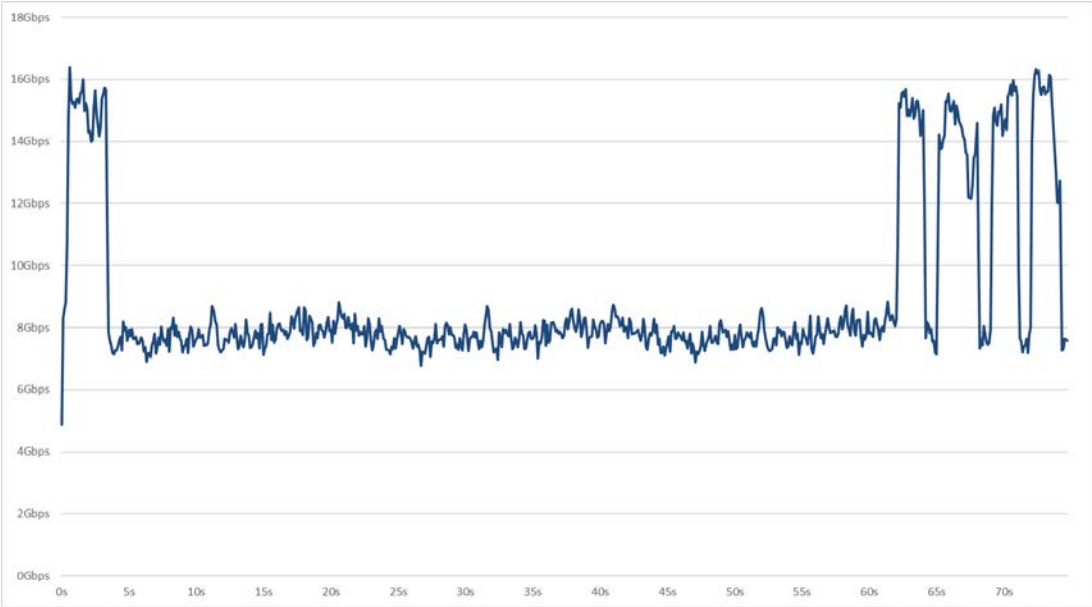
11.3. Eredmények

Minden egyes támadásfajtára 10 tesztet készítettem, mivel a rendszer időváriáns. A tesztek olyan eredményeket hoztak (36. ábra), amik megegyeztek a várakozássokkal. Minél nagyobb a támadás csomagsebessége, annál gyorsabb az észlelése. Látható, hogy az igen lassú #5 támadás is átlagosan 17ms alatt észlelve lett.



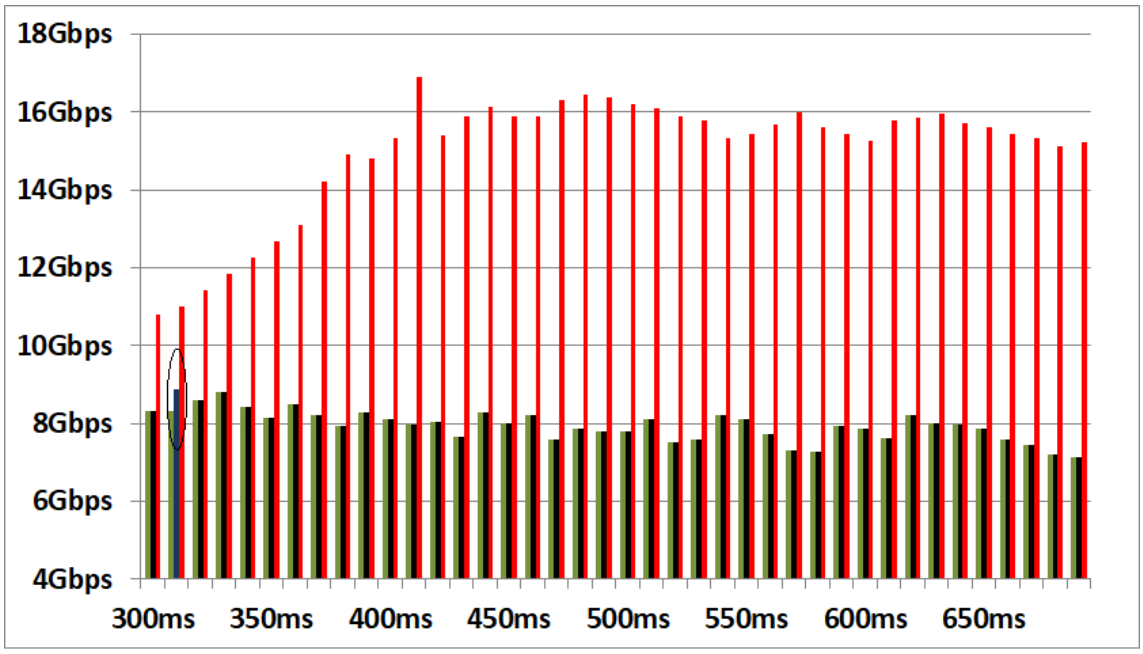
36. ábra A detekciós idők

Az FPGA-ba ezúttal egy statisztikai modul is került, hogy vizualizálni lehessen a kiszűrt csomagok számát. Az #1 támadás detekciós eredményei alapján készült a 37. ábra, 38. ábra és a 39. ábra. Az ábrákon látható, hogy a támadás tüskéit gyakorlatilag nyom nélkül eltünteti a detektor.



37. ábra A DNS támadás normális forgalommal való keverésének aggregáltja

38. ábra A támadás produktuma



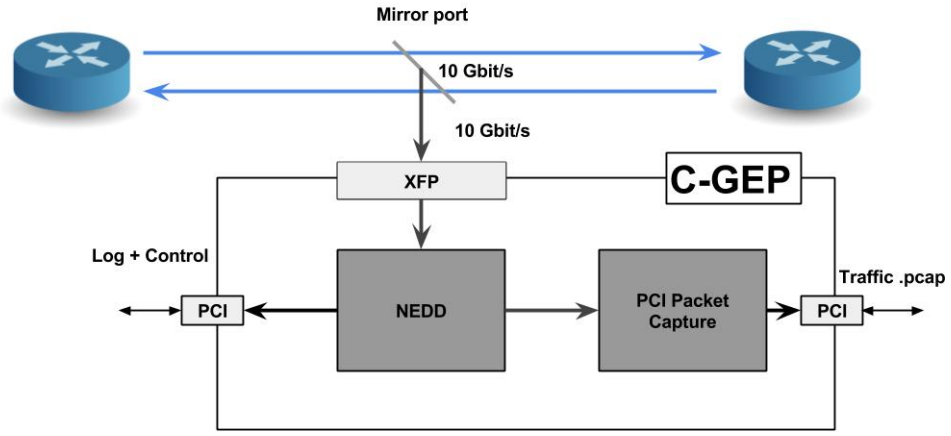
39. ábra A támadás felfutásának kinagyítása. A piros oszlop az aggregált forgalom. A züld oszlop normál forgalom. A fekete oszlop az átengedett forgalom.

12. A rendszer integrált real-time tesztelése

A rendszer tesztelésének következő logikus lépése a fizikai real-time tesztelés. Ebben a fejezetben kizárólag a detekciós tevékenységet teszteltem, a beavatkozást nem. A rendszert a NIIFI hálózatában helyeztem el tesztelésre, arra az FPGA-PC setupra installláltam, amin a teszt forgalmak rögzítése történt. Ebben a fejezetben tárgyalni fogom a két tesztelési módszert, amit alkalmaztam, a rendszer szükséges módosításait és az tesztelés eredményeit.

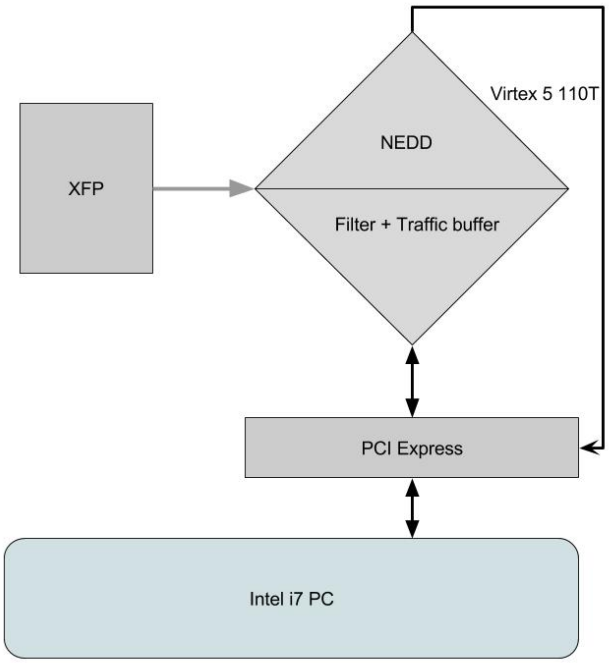
12.1. A teszteléshez szükséges módosítások

Ha teszteljük a detektort, akkor a helyes működés igazolásához szükség van gerjesztésre (x-re) és az eredményekre (y-ra), valamint a rendszer funkcionalitásának transzparens modellezésére. Az eredmények előállítása adott. A gerjesztés rögzítésére nincs megoldás jelenleg, vagyis, olyan nincs, ami működik NEDD-del integrálva.



40. ábra Rendszer blokk diagramja

Viszont az ipari konzulensem által megírt capture készítő FPGA firmware és PC software rendelkezésre áll. Adja magát a lehetőség, hogy NEDD-t integráljuk a meglévő forgalomrögzítő rendszerrel. Ezt a 40. ábra Rendszer blokk diagramjaés a 41. ábra mutatja, ami alapján beláthatjuk, hogy a gerjesztést mind a forgalomrögzítő mind a detektor megkapja. A log készítés és a forgalommentés a PC harddrive-ra történik, ami 4 WD 1TB Caviar Black Raid0-ba kötéséből jött létre, biztosítva a nagy rögzítési sebességet.



41. ábra A teszt rendszer fizikai vázlata

Ezzel a módosítással el is készült a gerjesztés (x), és az eredmény (y) rögzítésére alkalmas berendezés.

12.2. Tesztelési módszerek

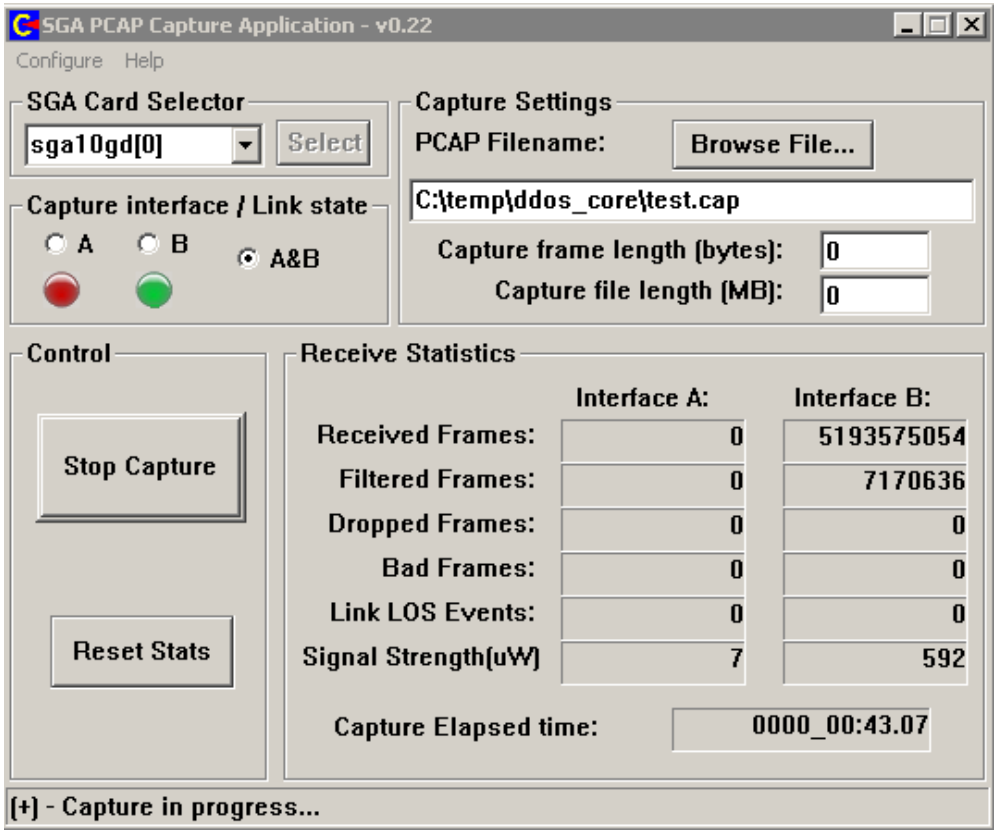
A rendszer tesztelésére több módszert lehet alkalmazni. Fontos volt kiválasztani azokat, amik jó eredményeket hoznak, és nem igényelnek hatalmas erőforrásokat.

1. Módszer: C++ model készítése NEDD szimulálására eredmények összehasonlítása.
   Előnyök: A helyes, specifikáció szerinti működést bizonyítja.
   Hátrányok: A használhatóságot nem bizonyítja, hosszú C++ modell fejlesztési idő a párhuzamosított logika miatt, hosszú futási idő precíz modell esetén. Nagyon nagy mennyiségű (1-10TB) forgalom rögzítését igényli.
2. Módszer: Speciális forgalom generálása, amivel a specifikáció elemei megvalósításának határértékeit lehet kimérni.
   Előnyök: A helyes, specifikáció szerinti működést bizonyítja. Rendelkezésre állnak az eszközök hozzá. Tesztelés viszonylag gyors.
   Hátrányok: Kevésbé megbízható, mint az 1. módszer.
3. Módszer: A detektált támadások rögzítése és manuális validálás a támadásokon.
   Előnyök: Egyszerű, több fejlesztést nem igényel. Kevesebb forgalmat kell elmenteni.
   Hátrányok: Csak a fals pozitív hibákat lehet észrevenni.
4. Módszer: A helyes működés igazolása benchmark-al.
   Előnyök: Nagy pontosság, megbízhatóság, a konkurenciához képest pozícionálja a terméket.

Hátrányok: Drága, hardware eszközök beszerzése nehézkes. A mirror-on csonkolt csomagokat kap a rendszer biztonsági okok miatt is, a benchmark rendszernek képesnek kell kezelnie az ilyen csomagokat. A rendszert csak a másik rendszerhez tudjuk mérni, ha alacsony minőségű a rendszer nem lesz objektív az eredmény.

A teszteléshez az második és harmadik módszert választottam mivel ez a két módszer jól kiegészíti egymást és jól lefedi a tesztelendő feladatokat.

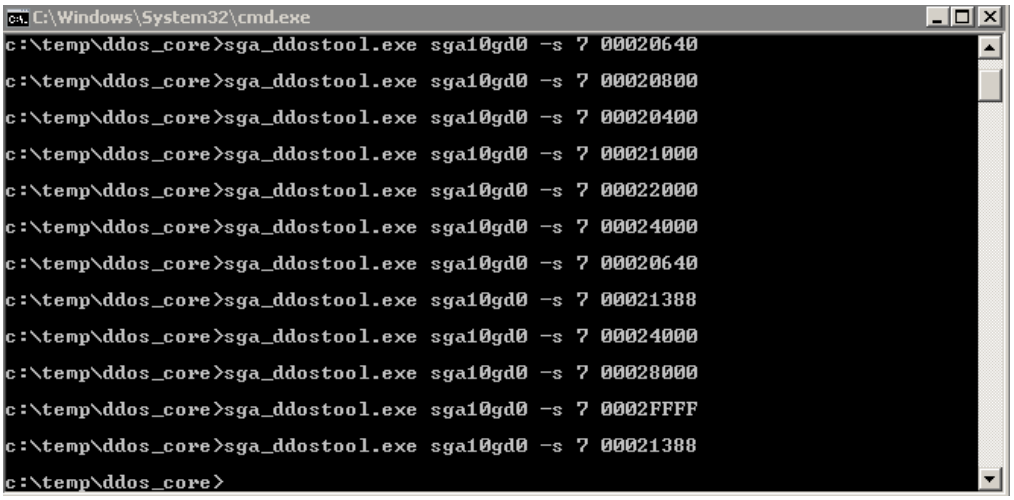
12.3. A második tesztelési módszer eredményei



42. ábra A forgalommentő program kezelői felülete

A tesztrendszer két darab FPGA kártyából áll, egyiken van az ismertetett teszt rendszer, a másikon egy precíziós forgalom generátor. A 42. ábra és a 43. ábramutatja a NEDD konfigurációs interface-eit.

A rendszer paramétereinek határértékeit sikeresen kimértem.

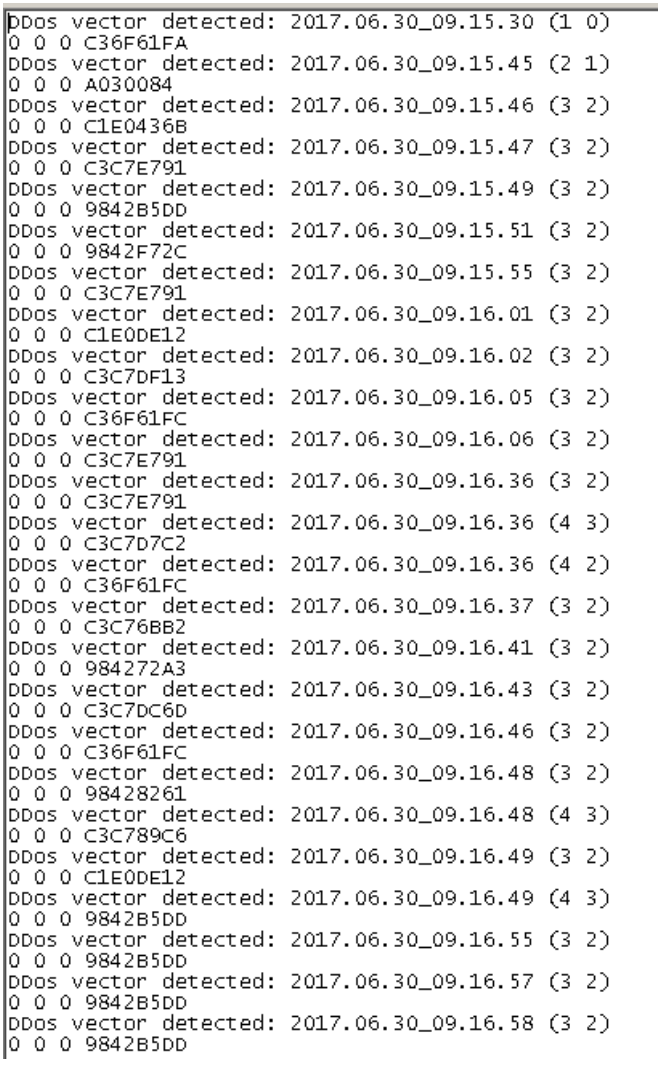


43. ábra A NEDD konfigurációs interface

12.4. A harmadik tesztelési módszer dokumentálása

A rendszert munkatársammal telepítettük a NIIFI szervertermében, az FPGA chipre ugyanaz a firmware került, amit a második teszthez is használtam; és a a konfigurációs eszköztár is a második tesztnél bemutatott volt.42. ábra A forgalommentő program. A különbség annyi, hogy az eredményeket egy állományba írja, ahogyan a 44. ábra szemlélteti.(Az ábrán teszt adatok vannak).

A rendszer teszteléséhez hosszabb szünet nélküli időre bekapcsolva hagytam a berendezéseket és pcap-t kiértékeltem a végén a loggal együtt.



44. ábra A rendszer naplója

12.4.1. Eredmények kiértékelése

A minta gyűjtés kezdete 2017.06.30., a minta gyűjtés vége 2017.8.31.. A detektor 1Gbps trigger feltétellel volt kalibrálva, azaz 1Gbps bitsebességnél nagyobb forgalmú end-pointokat vizsgálta. A tesztelési időben 8 külön álló ilyen esemény volt, ebből 6 reflektált támadás volt. A másik kettő rövid lefolyású UDP anomáliák voltak. A 6 reflektált támadásnál biztosan kijelenthetjük, hogy ezek DDoS támadás részei voltak. Az első UDP támadásnál nem lehet ilyen nagy biztonsággal megállapítani, hogy támadás-e. Az első UDP támadás csak 6ms-ig tartott, a csomagok megvizsgálásából támadásra lehet következtetni (klónozott payload). A második támadás 2s-ig tartott, a csomagok megvizsgálásából normál legális adatforgalomra lehet következtetni.

13. Rendszerintegráció

A NEDD önmagában nem túl hasznos, mivel nem tartalmaz beavatkozó szervet és nem tud védelmet biztosítani a kibertámadások teljes repertoárja ellen. Ebben a fejezetben két-fajta beavatkozó szerv integrálását, valamint egy software-es IDS rendszer hipotetikus integrációját mutatom be.

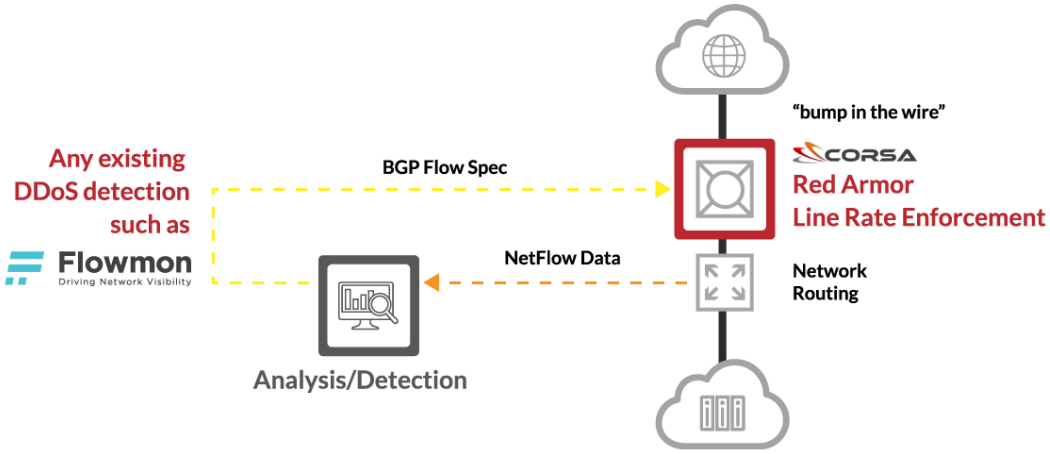
13.1. Switch-be épített szűrés alkalmazása

Mivel a legtöbb switch tartalmaz szűrési funkciókat, ezért mindenképp érdemes megvizsgálni annak a lehetőséget, hogy a hálózat switch-eit használjuk beavatkozó szervként.

A 45. ábramutatja be a switch vezérlés blokkvázlatát. A switch control port-ján küldi ki az új vezérlőüzeneteket, amit átvesz a switch és alkalmazza az üzenetben található szabályokat.

13.2. Bump-in-wire „Network Security Enforcement” integrálása

A NEDD DDoS detekciós ezközt direkt támadások megállítására tervezet szűrök vezérlésére is lehet használni. Ebben a pontban a Corsa NSE7000 „Red Armor” integrálását mutatom be [25]. A NSE7000 direkt arra lett tervezve, hogy bármilyen IDS rendszer vezérelhesse.



45. ábra NSE7000 integrálásának blokk vázlata

A 45. ábra mutatja, be hogyan lehet a „Red Armor”-t elhelyezni a rendszerbe. Mivel a „Red Armor” direkt támadások megállítására lett kifejlesztve, ezért lényegesen könnyebb integrálni, mint egy átlagos switch-et.

A NSE7000 három egyszerű lépésben integrálható.

1. A RA behelyezése az edge-router elé.

2. A NEDD felkészítése BGP Flow Spec vagy OpenFlow formátumú üzenetek küldésére, NEDD kimeneti üzenetei tartalmilag tökéletesek, csak a szintaktikát kell átalakítani
3. A szabályozási kör bezárása, switch rákötése NEDD-re, NEDD bekötése a RA-ba

13.3. Software IDS rendszer integrálása NEDD-hez

FPGA-n nehéz bizonyos (többnyire layer 7) támadások elleni védelmet megvalósítani, a limitált memória miatt. A fizikai erőforrások erősen korlátozottak, ezért célszerű a rendszert PC-n futó software komponenssel is kiegészíteni. A hardware biztosítja a rendszer teljesítményét, a software biztosítja az intelligenciát és flexibilitást.

A közelmúltban rengeteg IDS software került forgalomba különböző funkcionalitással, de egyik sem tud feldolgozni adatot azzal a sebességgel, amire szükség lenne ahhoz, hogy egy egy teljes adatközponti, vagy IP exchange hálózatot kiszolgáljon. Ha IDS software-t szeretnénk csatlakoztatni a rendszerhez, szükség van valami fajta hardware-es gyorsításra, például FPGA alapú csomagszűrési eljárásra.

Többfajta filterezési eljárást kidolgozhatunk, csökkentve a software-re jutó feldolgozandó adat mennyiséget. Filterezhetünk támadás típus szerint: ha arra vagyunk kíváncsiak van-e folyamatban HTTP GET támadás, akkor a *nem* http GET csomagokat kiszűrhetjük a software-nek továbbított adatfolyamból. Filterezhetünk lokalitás szerint: honnan/hová megy a csomag. Ezeket a szűrési módokat időosztásos módszerrel célszerű váltogathatjuk.

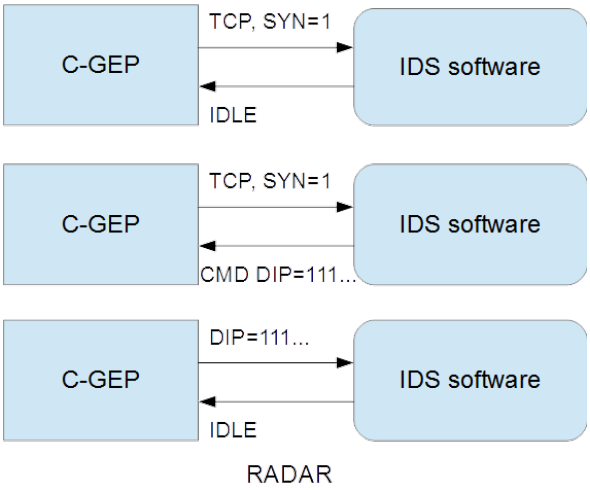
Mivel a software az egész adatfolyamot nem tudja egy adott időpontban átlátni, ezért fontos, hogy valamilyen hatás-mechanizmussal vissza tudjon jelezni a FPGA-nak, hogy milyen adatra van szüksége.

A RADAR lényege, hogy a software-alapú IDS rendszer képes visszajelezni a NEDD-nek: hogyan változtassa az átadandó adatai időbeli vagy térbeli felbontását, az általa megkapott adatok mely szegmenségre „világítson rá” és adja tovább a software-es IDS-nek.

13.4. RADAR működés elvi példa

A 46. ábra egy példát mutat be egy multivektoros támadás feltérképezésére. Ennek menete a következő.

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



46. ábra Példa a RADAR működésére

13.5. Filterezés támadásfajta szerint

A következő felsorolás szűrési feltételeket ad a leggyakoribb DoS támadásokra.

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.

TCP FIN attack

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

TCP RESET attack

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

UDP Fragment attack

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

UDP Flood attack

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

Slowloris attack

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

ICMP flood attack

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

HTTP get attack

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

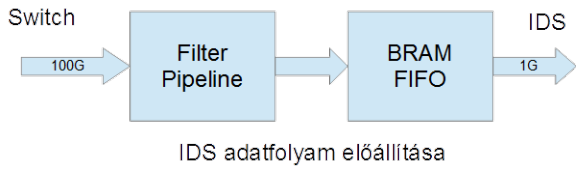
HTTP post attack

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

13.6. 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.

13.7. Adatfolyam előállítása az IDS software számára

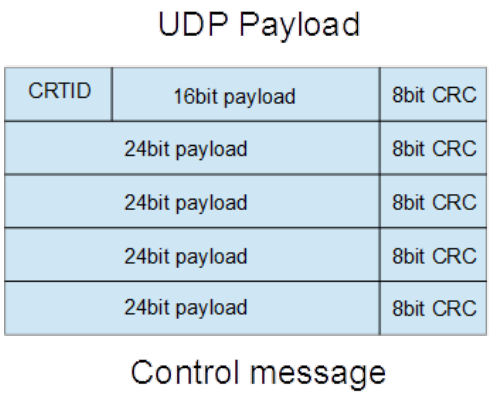


47. ábra IDS adatfolyam előállítása

A fenti blokkvázlat (47. ábra) az IDS software-be tartó adatfolyam létrehozását mutatja be. A switch-től 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.

13.8. 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 a software-es IDS-nek továbbadandó streamből. 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. Az ilyesfajta hibák minimalizálása érdekében kénytelenek vagyunk nagy mennyiségű redundanciát beleépíteni az üzenetbe.

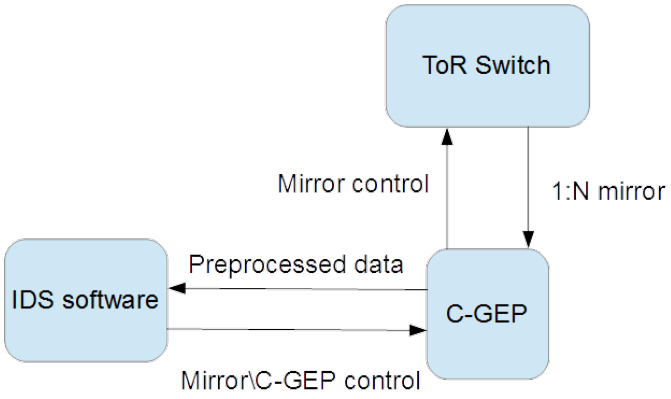


48. ábra Vezérlő üzenet, amit a software IDS-nek küld a NEDD

A fenti 20 byteos üzenet egy példa a redundáns vezérlő üzenetre (lásd 48. ábra). Az UDP fejléc port-jaival tudjuk jelezni, hogy vezérlő üzenet jön, ezt ki kell egészíteni egy előre meghatározott CRTID-vel ami jelzi, hogy tényleg vezérlő üzenet érkezett.

13.9. A switch vezérlése

A RADAR elvét kiterjeszthetjük a switch vezérlésére is. A mirror nem ideális, azaz nem tud minden egyes csomagot – ami a switchen áthalad – teljes terjedelmében kimásolni. Ha emellett képesek vagyunk vezérelni a mirrort, akkor az előbb bemutatott mechanikát kiterjeszthetjük a switchre is. Ez esetben tehát a switch mintavételezését vezéreljük az IDS rendszerrel, a 49. ábra szerinti elrendezésben.



49. ábra RADAR switch vezérléssel

A software nem csak a C-GEP-t szabályozza, hanem a mirror mintavételezését is. Ezzel a funkcióval csökkenthetjük a behatolás-detekciós rendszer által a switch-ben leforglalt erőforrásokat (például a mirror sávszélességét).

14. Támadások megállítása

Az eddig bemutatott rendszer kiválóan alkalmas támadások észlelésére, azaz IDS feladatok ellátására. Viszont ha IPS (Intrusion Prevention, behatolás-megelőzés) feladatokat is szeretnénk ellátni, akkor valamiféle extra funkcionalitásra is szükség lesz. Sajnos nem létezik olyan beavatkozás, ami minden egyes támadásformához megfelelő lenne, ezért érdemes ezeket külön fejezetben tárgyalni. Itt csak a kilenc leggyakoribb támadáshoz tartozó beavatkozást fogom tárgyalni, valamint csak azokat a beavatkozás formákat, amiket az „Az FPGA alapú védelmi rendszer ” című fejezetben tárgyaltam.

14.1. Direkt spoofolás-mentes támadások

Ebbe a kategóriába esnek a SYN, UDP, UDP frag, ACK flood-ok spoof-olatlan verziói. Ezek a támadások körülbelül az összes támadás 40% teszik ki. Ezekre a támadásokra a legegyszerűbb választ adni. Az IPS feladata ebben az esetben az, hogy a támadók IP címét kikódolja. Ezek a támadás detektálása utáni egyszerű flow-onkénti sebesség osztályozással kiszűrhetők a legtöbb esetben. Egy DDoS támadásnál minden flow lassul, ami nem támadó flow, például a congestion control miatt.

Ha az IPS kikódolta a támadók IP címét, azokból a címekből ACL szabályokat készít, amit elküld a switchnek. Az ACL szabályok vagy letiltják a megadott címeket a switchben, vagy korlátozzák a throughput-jukat.

14.2. Direkt spoofolt támadások

Ebbe a kategóriába esnek a SYN, UDP, UDP frag, ACK flood-ok spoof-olt verziói. Ezek a támadások becslésem szerint (sajnos pontos számok erről nem állnak rendelkezésre) 5-10%-t tehetik ki az összes támadásnak. Ezen kategóriára a legnehezebb ebben az elrendezésben jó beavatkozó jelet találni. Az IP alapú filterezés sajnos nem lehet megoldás.

Sajnos a legtöbb esetben nem lehet sok mindent tenni ebben a topológiában a normális forgalom veszélyeztetése nélkül.

14.3. Reflektált erősített támadások

Ezek a támadások teszik ki az összes támadás ~50%-át. Ebbe a kategóriába tartoznak az NTP, DNS, SSDP, CHARGEN, RIP támadások. Ezek ellen a támadások ellen sem lehet IP filterezést használni, mert a támadó üzenetek legális szolgáltatóktól jönnek, akiket nem lehet letiltani. Ezzel szemben tartalom alapú szűrést be lehet vezetni.

Példaképpen: NTP támadás esetén a támadott IP címére érkező (NTP mon_getlist response) üzeneteket le lehet korlátozni.

A többi reflektált és erősített támadásnál ezt a mintát kell követni, azaz a támadott IP címekre érkező adott támadásokra vonatkozó respone csomagokat letiltja.

15. Összefoglalás

A jelenlegi kiberbiztonsági folyamatok komoly kihívások elé állítják a szakértőket. A hackerek olyan támadási módokat találtak, amiket jelenlegi mainstream rendszerek nem csak, hogy nem tudnak megakadályozni, de gyakran nem is tudják érzékelni. A gyors lefolyású úgynevezett Hit-and-Run támadások egyre elterjedtebbek. Egy ilyen támadás egy tüskéje akár egy másodpercnél kevesebb idő alatt is nagyon komoly károkat tud okozni. Egy ilyen tüske ma már a kiterjedt IoT botnet hálózatoknak köszönhetően kis anyagi és szaktudás befektetésével könnyen elérhet több tíz Gigabit/s bitsebességet is. Az általam fejlesztett FPGA alapú megoldás ezekre a problémákra ajánl megoldást.

A dolgozatomban bemutattam a legjelentősebb DDoS támadásformákat, valamint az ezek ellen használt védelmi megoldásokat, azok előnyeit, hátrányait, valamint azokat a legfrissebb trendeket, amik ezt a területet jellemzik. Szakértők számára egyre világosabb, hogy változtatni kell a jelenlegi paradigmákon ahhoz, hogy versenyképesek maradjunk a hackerekkel.

Megterveztem egy hardware alapú DDoS detektort, ami a támadások 95%-val képes elbánni. A detektor tervezése során nem csak a funkcionalitásra, hanem a felhasználhatóságra is figyeltem, célom volt egy olyan eszköz elkészítése, ami könnyen behelyezhető bármilyen hálózatba és ott jól tud működni. A munka detekció részére összpontosítottam, hogy minél jobb megoldást tudjak mutatni.

Az általam tervezett és fejlesztett DDoS detektor implementációja sikerrel zárult, egy működő hasznos rendszert hoztam létre, ami ezred másodpercek alatt képes a különféle típusú, L3/4 szintű DDoS támadások észlelésére.

Az általam fejlesztett, FPGA-alapú detektor ezredmásodperces reakció-idővel tudja észlelni az UDP fragmentation, UDP, DNS, NTP, SSDP, SNMP, SYN, ACK, RIP, CHARGEN flood-okat., mindezt 100%-osan viselkedési illetve heurisztikus alapon, azaz nem kell „belenéznie” a csomagok felhasználói tartalmába. Ezek a teszik ki napjaink DDoS támadásainak közel 95%-át.

A detektor három fő részből áll össze; egy dekodoló, parse-oló egységből, egy host sebesség osztályozó, és a magas szintű detekciós egységből. Az implementáció során számos alkalommal kellett olyan speciális mérnöki döntéseket hozni, amikre az FPGA-alapú fejlesztés lehetőségei (például a párhuzamos adatfeldolgozás), vagy korlátai (például a memória-limitációk) miatt volt szükséges.

A feldolgozási lánc első eleme a a dekodoló és parse-oló egység. Ez minden egyes csomagot átalakít egy-egy 15 byte-os adatponttá, ami könnyen kezelhetővé teszi a csomagokat. A host sebeség osztályozó kiválasztja az adott host-ra érkező forgalom alapján, hogy lehetséges-e az, hogy a host támadás alatt áll. Erre a lépésre azért van szükség, mert az FPGA-n belül nincs elég erőforrás az összes host folyamatos vizsgálatára. Az feldolgozási lánc utolsó fő modulja a magas szintű detekciós egység. Ez a detekciós egység 16 különböző algoritmust futtat párhuzamosan, ezzel észleli a fenti támadásokat.

A detekciós algoritmusokat magam válogattam össze ezekhez a támadásokhoz; így, egyben sehol sincsenek publikálva – nemhogy implementációs, de tervezési szinten sem; FPGA-implementációra pedig egyáltalán nincs példa. Ezek az algoritmusok megtalálhatóak a dolgozatom 6.5 és 6.6 fejezeteiben.

A detektor implementációja végeztével sikerült megvalósítanom az általam megtervezett funkciókat. A verifikációt több részből állítottam össze. A detektort sikeresen szimuláltam az Isim-mel, nagy mennyiségű különböző forgalmat használva, ezzel ellenőriztem a koncepció helyességét. Mind moduláris mind funkcionális szimulációkat készítettem, hogy minél jobb minőségű VHDL kóddal kezdhessek bele a fizikai tesztelésbe. A tesztelés során igyekeztem minél valósághűbb környezetet teremteni, ezért valós támadások pontoson timestamp-elt felvételeit játszottam vissza az érzékelőnek – mind a szimuláció során, mind a hardware-es tesztek során. A tesztelést hardware-en folytattam, először 1Gbps majd 10Gbps összeállításban teszteltem. Valódi támadások forgalmát használva teszteltem UDP, DNS, NTP, SYN, RIP, UDP fragmentetation támadásokra.

A fizikai tesztek bebizonyították, hogy még a leglassabban detektált támadások is ezredmásodperces nagyságrendű idő alatt észlelhetők. Ilyen reakció-időre nem találtam példát a vonatkozó irodalomban, pedig ezzel számos, rövid fel- és lefutású támadás, vagy tüskejellegű támadássorozat kiszűrhető.

A detektort a jövőben szeretném tovább fejleszteni úgy, hogy a legújabb trendekre felkészítsem, és fejlesszem a képességeit. Az éles tesztelés folytatására is szükség van lehetőleg most már egy nagyságrendekkel nagyobb adatközpontba vagy IPX-be kihelyezve, mivel valós képet csak és kizárólag egy, a jelenlegi, három hónaposnál hosszabb próbaüzem után kaphatok a detektor képességeiről.

.

Irodalomjegyzék

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

[2] Akamai, State of the internet, 2016 4. negyedév: https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/akamai-q4-2016-state-of-the-internet-security-report.pdf

[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.: https://www.incapsula.com/ddos/attack-glossary/udp-flood.html

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

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

[7] USA federal goverment, A legjellemzőbb reflektált, erősített támadás típusok: https://www.us-cert.gov/ncas/alerts/TA14-017A

[8] Brian Krebs, Krebs on security, Krebs elleni támadás.: https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/

[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.: https://www.researchgate.net/publication/281374945_100_Gbits_network_monitoring_with_on-the-fly_reconfigurable_rules_for_multi-encapsulated_packets

[12] Akamai, Client Reputation.: 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] Internet engineering taskforce, IETF.org, RFC864.: https://tools.ietf.org/html/rfc864

[15] Anna Sempai, Hackforums.net, Mirai botnet.: https://hackforums.net/showthread.php?tid=5420472

[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.: 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] Infosec Institute, LOIC (Low Orbit Ion Cannon) – DOS attacking tool.: http://resources.infosecinstitute.com/loic-dos-attacking-tool/
[20] Xilinx, ISE design suite,.: http://www.xilinx.com/products/design-tools/ise-design-suite.html
[21] Brian Krebs, krebsonsecurity.com, Sony IoT exploit.: https://krebsonsecurity.com/2016/12/researchers-find-fresh-fodder-for-iot-attack-cannons/
[22] FPGA Networking, C-GEP ismertetése.: http://www.fpganetworking.com/160616/c_gep/index.html
[23] Abraham Wald, Google books, Sequential analysis.: https://books.google.hu/books/about/Sequential_Analysis.html?id=oVYDHHzZtdIC&redir_esc=y
[24] P. Dabholkar, A low latency asynchronous Jenkins hash engine for IP lookup.:
[25] Corsa, NSE7000.: https://www.corsa.com/security/ddos-mitigation/
[26] FlowMon, FlowMon DDoS Defender Webinar 2015.05.07.: https://www.youtube.com/watch?v=8WACBeXTmMQ
[27] FlowMon, FlowMon Probe.: https://www.flowmon.com/en/products/flowmon/probe
[28] Corsa, Corsa DDoS Whitepaper part 2.: www.corsa.com
[29] Corsa, Corsa NSE7000 Data sheet.: www.corsa.com
[30] Fortinet, FortiDDoS data sheet.: https://www.fortinet.com/content/dam/fortinet/assets/data-sheets/FortiDDoS.pdf
[31] Fortinet, FortiDDoS Handbook.: http://docs.fortinet.com/uploaded/files/3680/FortiDDoS_4.3.1_Handbook.pdf
[32] Cisco, NCS5500 Data sheet.: http://www.cisco.com/c/en/us/products/collateral/routers/network-convergence-system-5500-series/datasheet-c78-736270.html
[33] Mitko Bogdanoski; Shuminoski, Tomislav; Risteski, Aleksandar. International Journal of Computer Network and Information Security; Hong Kong5.8 (Jun 2013): 1-11.; Analysis of the SYN Flood DoS Attack.: http://www.mecs-press.org/
[34] Samad S. Kolahi, Kiattikul Treseangrat, Bahman Sarrafpour, Department of Computing, Unitec Institute of Technology, Auckland, New Zealand; Analysis of UDP DDoS Flood Cyber Attack and Defense Mechanisms on Web Server with Linux Ubuntu 13 http://unitec.researchbank.ac.nz/bitstream/handle/10652/3613/Analy-

sis%20of%20UDP%20DDoS%20Flood%20Cyber%20Attack%20and%20Defense%20Mechanisim%20on%20Web%20Server%20With%20Linux%20%281%29.pdf?sequence=1&isAllowed=y

[35] Taming the 800 Pound Gorilla: The Rise and Decline of NTP DDoS Attacks Jakub; Czyz Michael, Manaf Gharaibeh, Christos Papadopoulos, Michael Bailey, Manish Karir; California State University.: http://s3.amazonaws.com/academia.edu.documents/34662428/imc14_ntp_ddos.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1498740469&Signature=8HAErtkzFXRUxrZIzMbnEL%2BQlGQ%3D&response-content-disposition=inline%3B%20filename%3DTaming_the_800_Pound_Gorilla_The_Rise_an.pdf

[36] Amplification Hell: Revisiting, Network Protocols for DDoS Abuse, Christian Rossow, VU University Amsterdam, The Netherlands.: https://dud.inf.tu-dresden.de/~strufe/rn_lit/rossow14amplification.pdf

[37] Distributed Denial of Service: Taxonomies of Attacks, Tools and Countermeasures, Stephen M. Specht, Ruby B. Lee, Princeton University.: https://www.researchgate.net/profile/Ruby_Lee/publication/220922510_Distributed_Denial_of_Service_Taxonomies_of_Attacks_Tools_and_Countermeasures/links/0fcfd50bd9fda81b51000000.pdf

[38] Pál Varga, Georgios Kathareios, Ákos Máté, Rolf Clauberg, Andreea Anghel, Péter Orosz, Balázs Nagy, Tamás Tóthfalusi, László Kovács, Mitch Gusat, IEEE CNSM, Real-Time Security Services for SDN-based Datacenters.:

[39] Balázs Nagy, Péter Orosz, Tamás Tóthfalusi, László Kovács, Pál Varga, IEEE ICC, Detecting DDoS Attacks within Milliseconds by Using FPGA-based Hardware Acceleration.: beadva

[40] Corero, http://info.corero.com/rs/258-JCF- 941/images/corero-q1- 2017- ddos-trends- report.pdf

[41] Péter Orosz, Tamás Tóthfalusi, Pál Varga, “C-GEP: Adaptive network management with recon- figurable hardware,” in Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on. IEEE, 2015, pp. 954–959.