VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2020 Kyrylo Shpak

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

ZÁTĚŽOVÝ TESTER
STRESS TESTER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Kyrylo Shpak**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Václav Zeman, Ph.D.**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Kyrylo Shpak ***ID:*** 195432
Ročník: 3 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Zátěžový tester

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je vytvoření dílčích modulů určených pro zátěžový tester. Zátěžový tester je zařízení postavené na SW JMeter a představuje komplexní systém pro testování informační a komunikační infrastruktury. V první fázi je úkolem ověřit funkčnost a nalezení chyb uvedeného systému a provést stručný popis DoS útoků s důrazem na nové typy komunikačních protokolů, které útoky využívají. Hlavním výstupem práce je rozšíření systému zátěžového testeru o moduly určené pro testování pomocí amplifikačních útoků.

DOPORUČENÁ LITERATURA:

[1] HALILI, Emily H. Apache JMeter: A practical beginner's guide to automated testing and performance measurement for your websites. Packt Publishing Ltd, 2008.

[2] ERINLE, Bayo. Performance Testing with JMeter 2.9. Packt Publishing Ltd, 2013.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 8.6.2020

Vedoucí práce: doc. Ing. Václav Zeman, Ph.D.

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Bakalářská práce se zabývá problematikou zátěžového testování. Cílem práce je rozšíření nástroje JMeter o amplifikační moduly DoS útoků. Začátek práce uvádí do problematiky útoků typu DoS, spolu s popisem vybraných útoků. Jedním z bodů práce je ověření funkčnosti a nalezení chyb zátěžového testeru, jenž je komplexní systém pro testování informační a komunikační infrastruktury postavený na bázi aplikace JMeter. Poslední kapitola práce se věnuje implementaci DoS útoků.

KLÍČOVÁ SLOVA

DoS, DDoS, JMeter, Java, QUIC, SSDP, amplifikace, netsniff-ng, trafgen, zátěžové testování

ABSTRACT

The bachelor thesis deals with the issue of stress testing. The aim of this work is to extend the JMeter tool with amplification modules of DoS attacks. The beginning of the work introduces the issue of DoS attacks, along with a description of selected attacks. One of the points of the work is to verify the functionality and find errors of the stress tester, which is a comprehensive system for testing the information and communication infrastructure based on the JMeter tool. The last chapter deals with the implementation of DoS attacks.

KEYWORDS

DoS, DDoS, JMeter, Java, QUIC, SSDP, amplification, load testing, netsniff-ng, trafgen

SHPAK, Kyrylo. *Zátěžový tester*. Brno, 2020, 55 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: doc. Ing. Václav Zeman, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Zátěžový tester“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu doc. Ing. Václavu Zemanovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

S rostoucím zlepšením informačních a komunikačních technologií kladou uživatelé větší důraz na kvalitu poskytované služby. Mezi kritéria kvality patří rychlost odezvy, výkonnost a dostupnost požadované služby. Po každém nasazovacím cyklu aplikace by měl probíhat náležitý výkonnostní test. Jedním ze způsobů měření výkonnosti informačně technologické infrastruktury je zátěžové testování. Zátěžové testování pomáhá odhalit potenciální nedostatky a problémová místa služby. Kvalitní služba musí být dostatečně odolná vůči všem scénářům jejího použití. Velkou hrozbu dnes představují kybernetické útoky, kvůli kterým firmy nesou nejen velké ztráty, ale také poškození své reputace. Pokles výkonnosti pod přijatelnou úroveň se odráží na návštěvnosti služby jejími zákazníky. Způsobit tento pokles, nebo úplně znepřístupnit službu legitimním uživatelům, může útok typu DoS (Denial of Service). Emulace tohoto typu útoku ve své síti pomáhá lépe porozumět jak se proti takovýmto útokům efektivněji bránit.

Tato práce je zaměřena na rozšíření zátěžového testeru o moduly DoS útoků. Dalším cílem je ověření funkčnosti a nalezení chyb zátěžového testeru, jenž je zařízení postavené na aplikaci JMeter a představuje komplexní systém pro testování informační a komunikační infrastruktury. Systém obsahuje implementaci různých typů DoS útoků a umožňuje monitorování jejich průběhu.

První kapitola této práce popisuje jednotlivé typy DoS útoků a techniky jejich provedení. Jedním z bodů práce je zaměření se na nové typy komunikačních protokolů. V této práci byl vybrán komunikační protokol QUIC (Quick UDP Internet Connections) a byl analyzován z hlediska bezpečnosti vůči DoS útokům.

Na začátku druhé kapitoly práce je stručný úvod hlavních metrik zátěžového testování. Dále je v této kapitole seznámení s aplikací JMeter a seznámení s nástrojem Trafgen, který je jádrem DoS útoků, sloužící ke generování zátěže. Závěrem druhé kapitoly je testování systému a znázornění výsledků testů.

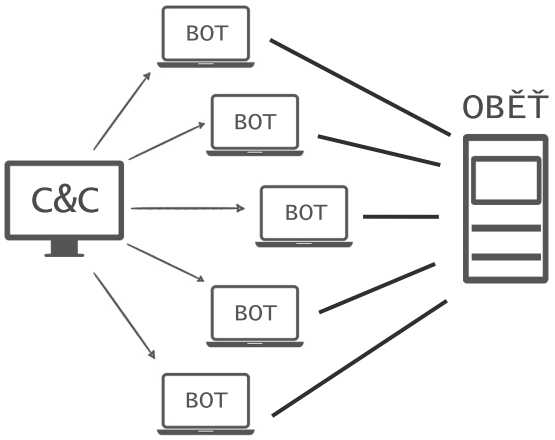
Třetí kapitola práce se věnuje rozšíření aplikace JMeter amplifikačními moduly. Na začátku kapitoly je popsána konfigurace vývojového prostředí a také podrobný návod jak se vytváří přídavné moduly aplikace JMeter. Závěrem kapitoly je realizace a testování amplifikačních modulů.

1 DoS útoky

S rostoucím počtem zařízení připojených k internetu vidíme kolem sebe vznik mnoho nových technologií a rozšíření použití již existujících. Firmy více reportují výskyt DoS (Denial of Service) útoků ve svých sítích. Cílem DoS útoku je odepření služby. Většinou se jedná o cílený útok, kdy se útočníci snaží omezit přístup k určitým službám na síti nebo úplně znemožnit jejich využití legitimním uživatelům. Firmy napadené tímto typem útoků nesou velké ztráty, někdy můžeme mluvit o stovkách tisíc dolarů za hodinu. Podle zpráv společnosti CloudFlare, která se specializuje také na ochranu vůči DoS, počet útoků stále narůstá a jsou více sofistikované.

1.1 Distribuovaný útok

DDoS (Distributed Denial of Service) je distribuovaná varianta útoku DoS, ve které geograficky rozptýlené, útočníkovi podřízené, stroje tzv. boty, jsou použité proti konkrétnímu cíli za účelem odepření služby. Taková síť kompromitovaných strojů je zvaná botnet. Využitím výkonných schopností každého takového stroje se zvyšuje účinek prováděného útoku. Botnet vzniká nakažením cizího stroje počítačovými viry, trojským koněm a jinými typy malware. Nejjednodušším a nejpoužívanějším způsobem kontrolování botnetu je pomocí vzdáleného serveru (obr. 1.1), na který se bot připojuje a v pravidelných intervalech kontroluje instrukce co má dělat. Obecně takový server je znám jako C&C (Command and Control) Server.



Obr. 1.1: Botnet architektura.

Některé botnety komunikují distribuovaným, peer-to-peer způsobem, tím pádem nelze jednoznačně určit jediný bod odkud boty přijímají instrukce. Fungují obdobně jako DHT síť používána BitTorrent, kdy se kompromitované stroje navzájem najdou bez použití „orchestračního“ serveru [1, 2].

1.2 Klasifikace DoS útoků

1.2.1 DoS založené na protokolu

Cílem útoků DoS založených na implementaci protokolu je zapříčinit nedostupnost cíle využitím slabin protokolu. Primárně se zaměřují na využití slabosti na úrovni 3 nebo 4 OSI vrstvy. Ataky můžeme rozdělit na tři velké skupiny:

- založené na TCP (Transmission Control Protocol),
- založené na UDP (User Datagram Protocol),
- ostatní.

Podíl útoků využívající protokoly prvních dvou skupin je největší. Nejpopulárnější z nich jsou TCP SYN Flood a UDP Flood resp. Vedle toho zmíněné dvě skupiny liší se podle typu spojení, potom rozlišujeme spojové (connection-based) a nespojové (connectionless) typy. Spojové útoky vyžadují navázání spojení mezi klientem a serverem, které je vyžadováno pro zahájení útoku, nebo pro zneužití samotného mechanismu připojení. Nespojové útoky naopak zneužívají toho, že pro jejich provedení mezi útočníkem a serverem nemusí být spojení navázané. Všechny útoky využívající protokol UDP a ICMP jsou příkladem nespojových typů útoků.

1.2.2 DoS podle vrstvy

Každý protokol sítě přes který jsou prováděny útoky se odkazuje na určitou vrstvu síťového modelu ISO/OSI resp. TCP/IP. Proto můžeme útoky DoS klasifikovat podle vrstev modelu TCP/IP. Rozlišujeme útoky konané na:

- vrstvě síťového rozhraní,
- síťové vrstvě,
- transportní vrstvě,
- aplikační vrstvě.

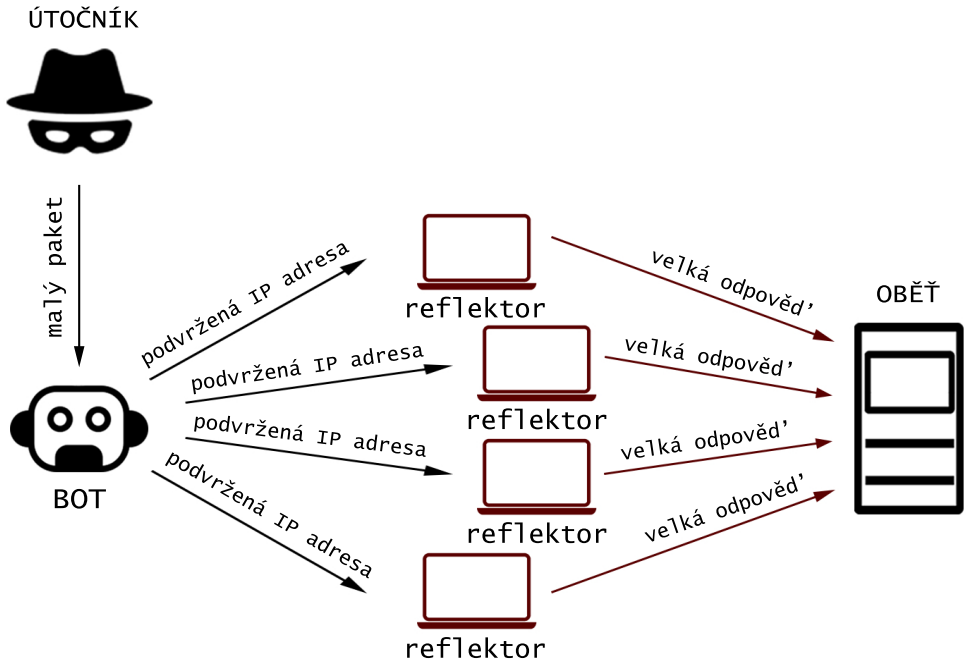
1.2.3 DoS podle mechanismu působení

Útoky záplavou paketů

Tento typ útoků se snaží přetížit šířku pásma komunikačního kanálu tím, že odesílá obrovské množství paketů s požadavkem na spojení nebo jiných irelevantních paketů. Na to aby pakety byly doručené k cíli a nezahozené firewallem se využívá dvou technik. První z nich je amplifikace neboli zesílení. Jedná se o techniku při které relativně nevelký za objemem paket dat vyvolává několikanásobně větší odpověď. Taková odpověď je směrována k cíli pomocí IP spoofingu, kdy útočník podvrhuje svoji zdrojovou IP adresu na adresu oběti. Nejvhodnějším příkladem útoků využívající amplifikaci je Smurf útok. Přibližný efekt amplifikace lze měřit pomocí BAF

(Bandwidth Amplification Factor). Čím je vyšší amplifikační faktor, tím větší účinek má útok a tím méně zdrojů potřebuje útočník. Zesílení jednotlivých útoků založených na UDP protokolu viz tabulka 1.1 (hodnoty převzaty z [3]).

Další technikou je reflexe neboli anglicky reflection, při které se používají protokoly typu výzva-odpověď. Reflektor je v tomto případě zařízení, které posílá odpovědi na zdrojovou IP adresu požadavku. Tato zdrojová IP adresa je podvržená pomocí techniky IP spoofing. Pro silnější efekt se využívá sada několika reflektorů. Příkladem reflektoru je DNS server.



Obr. 1.2: DDoS amplifikace s využitím reflektoru.

Když spojíme výše zmíněné dvě techniky s botnetem, vznikne masivní záplavový útok který bude obtížně detekovatelný, jelikož trafik procházející sítí se jeví jako legitimní, viz obrazek 1.2. Nejpopulárnější z útoků využívající těchto technik jsou: DNS, NTP, SNMP, SSDP amplifikace.

Útoky na odmítnutí služby

Když útoky záplavou paketů vyčerpávají síťové zdroje, tento typ je zaměřen na vyčerpání serverových zdrojů. Zaměřuje se především na zahlcení CPU, RAM paměti nebo diskového úložiště. Vytížením serverových zdrojů, může požadovaná služba zcela padnout a přestat obsluhovat legitimní požadavky. Využívá se zranitelnosti technologií které jsou „by design“ byly takhle navržené, avšak se používají pro provedení útoku.

Tab. 1.1: UDP amplifikační útoky.

Protokol	BFS
Memcached	10,000 až 51,000
NTP	556,9
CharGEN	358,8
CLDAP	56 až 70
DNS	28 až 54
SSDP	30,8
SNMP	6,3

Útoky na aplikační úrovni

Tento typ útoku se zaměřuje většinou na webové servery, jako například Apache nebo nginx. Detekce útoků na aplikační úrovni je více obtížná, hlavně z důvodu toho, že útočník se snaží napodobit chování legitimního uživatele. Pomocí protokolu HTTP (Hypertext Transfer Protocol) útočník sestavuje GET a POST dotazy tak aby pokrývaly co největší množství operací na straně serveru. Například dotaz na načtení informace z databáze, vrácení velkého souboru, kontrola správnosti uvedených údajů apod.

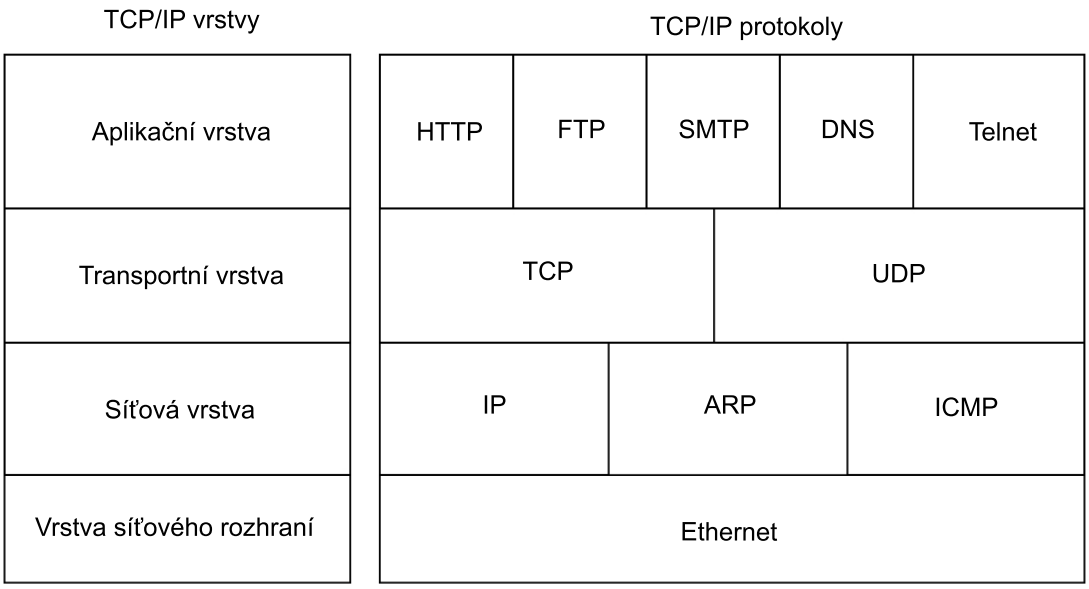
1.2.4 Multivektorový útok

Složitost mitigace DDoS útoků se zvětšuje s populárním trendem využití multivektorového útoku. Při tomto typu útoku jsou zapojené všechny výše uvedené techniky najednou. Šance úspěchu takového DDoS útoku je mnohem větší, tudíž když se jeden ze směru útoku nezdaří, druhý může být úspěšný. Podle statistiky společnosti A10, je multivektorový útok využit ve 75 procentech případů [4].

1.3 Bezpečnost TPC/IP modelu

TCP/IP (Transmission Control Protocol/Internet Protocol) je kolekce protokolů, které řídí komunikaci na Internetu. Celý model jako takový je soubor pravidel a postupů, který určuje jak komunikace v síti Internet bude probíhat. Znázornění TCP/IP modelu viz obrázek 1.3.

Navržený model klade důraz na spolehlivost sítí a odolnost proti výpadkům s možností automatické obnovy po selhání. S časem se ale začali objevovat mezery v bezpečnosti. Nejčastěji se zneužívá šesti TCP příznaků angl. „flags“ (SYN, ACK, URG, PSH, RST, FIN) které vyjadřují aktuální stav spojení. Například odesláním



Obr. 1.3: TCP/IP model, vrstvy a spojené s nimi protokoly.

paketu s příznakem RST na server klient informuje, že chce ukončit spojení. Zasláním velkého počtu paketů s příznakem RST útočník donutí server rozvázat spojení s uživateli. Pro provedení nežádoucího odpojení uživatele od serveru jsou využité další slabiny TCP/IP modelu:

- **odposlechnutí komunikace**: dovolí útočníkovi získat informace o použitých portech a pořadová čísla obou stran,
- **IP spoofing**: dovolí útočníkovi podvrhnout zdrojovou IP adresu spolu s kontrolním součtem IP datagramu,
- **zdrojové směrování (source routing)**: koncept, při kterém útočník určuje cestu kterou budou pakety zasílané.

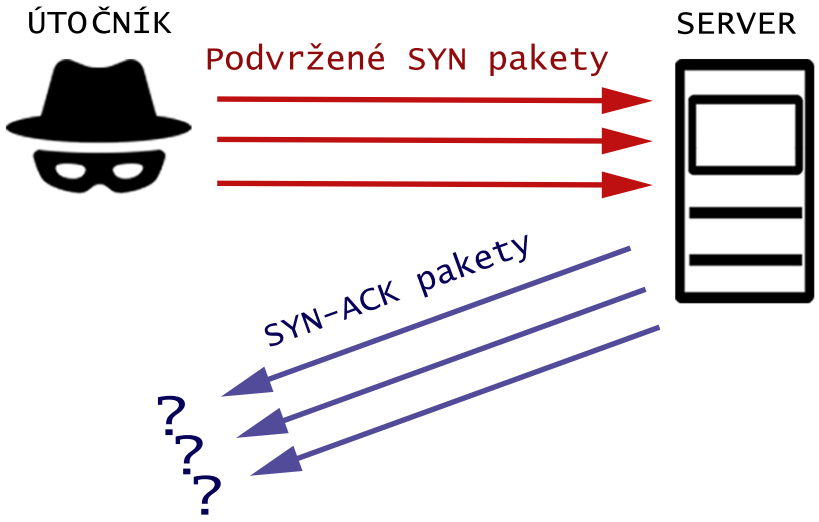
Dnes je možné některé z nedostatků TCP/IP modelu odstranit pomocí bezpečnostních technik. Stále se však potkáváme s případy, kdy jedna komunikující strana tyto techniky implementuje a druhá nikoliv.
V dalších sekcích jsou popsané jednotlivé DDoS útoky.

1.4 Popis konkrétních útoků

1.4.1 SYN flood

SYN flood útok zneužívá třícestný handshake (potřesení rukou) protokolu TCP. Třícestný handshake je zapotřebí, aby dvě strany mohly navázat spojení a skládá se ze tří kroků. Nejprve klient posílá zprávu SYN, která informuje server o požadavku

na spojení. Aby server dokázal odpovědět na požadavky jiných klientů, otevírá nové vlákno a alokuje systémové prostředky pro každého. Poté server odpovídá klientovi zprávou SYN+ACK, následně klient odesílá zprávu ACK, čímž je zahájeno nové spojení a komunikace může probíhat. Když klient hned v prvním kroku podvrhne zdrojovou IP adresu, server bude posílat odpovědi na neexistující adresu nebo adresu klienta který spojení nevyžadoval, viz obrázek 1.4. V každém případě se SYN+ACK



Obr. 1.4: Útok SYN flood s podvržením IP adresy odesílatele.

paket zahodí. Ještě po nějakou dobu server bude toto spojení držet v paměti a ukládat informace o jeho stavu. Pokud útočník zaplaví server SYN pakety, dojde k rychlému vyčerpání serverových prostředků, čímž zamezí ostatním uživatelům využití této služby tzn. způsobí její odepření.

1.4.2 UDP flood

Útok využívá vlastnost UDP protokolu, při které mezi klientem a serverem nemusí být navázané spojení. Útočník generuje velké množství UDP paketů s podvrženou IP adresou odesílatele, které jsou následně posílané na náhodné porty oběti, způsobující tím přetížení cílové sítě. Jakmile server obdrží UDP paket na určitý port, první co udělá, zkontroluje jestli nějaká aplikace naslouchá na tomto portu. Když žádná aplikace v tuto chvíli na daném portu nenaslouchá, server odpovídá ICMP (Internet Control Message Protocol) „Destination Unreachable“ paketem. Vyčerpání systémových prostředků může také podléhat zprostředkujícím entitám, jako je firewall nebo router. Tento typ záplavového útoku, UDP flood, se měří v Mbit/s (Megabit za sekundu) pro šířku pásma a PPS (packets per second) [5].

1.4.3 Sockstress

Sockstress spadá do kategorií pomalých útoku nevyžadujících velkou šířku pásma. Na rozdíl od SYN flood útoku, spojení mezi útočníkem a serverem je zcela navázané. V posledním kroku kdy útočník odesílá ACK paket, nastavuje velikost okna na hodnotu 0. Tím říká serveru, že v daný okamžik nemůže přijímat data a server musí počkat. Server periodicky odesílá dotaz, jestli už může začít data vysílat. Útočník cílí na konkrétní typy systémových prostředků a jádra, jako jsou čítače, časovače a paměťové zásoby. Následkem útoku je zastavení TCP služby nebo vynucení restartování serveru, který po restartu může stále neodpovídat.

1.4.4 Slow HTTP GET/POST

Útoky aplikační úrovni. V obou případech se útočník snaží otevřít maximální počet spojení se serverem aby zabránil využití služby jiným uživatelům. Účelem útočníka je držet tyto spojení aktivními co nejdelší dobu. Metody `GET` a `POST` HTTP protokolu jsou běžně využívané, používají se pro načtení webové stránky (její stažení) a pro odeslání dat na server resp.
V případě metody `GET` útočník posílá neúplný HTTP požadavek. Server předpokládá, že uživatel má pomalé internetové připojení a čeká až dorazí zbytek HTTP GET hlavičky, jinak požadavek nemůže zpracovat. Útočník ale posílá zbytek hlavičky po částech a co nejpomaleji ve snaze obejít časovač vypršení spojení. Tento typ útoku je známý jako Slowloris.
V případě metody `POST` je zaslána kompletní HTTP hlavička, kde útočník definuje velký objem dat k odeslání v poli „Content-Length“. Následující odesílání těla zprávy probíhá pomalu a po malých částech, což vynucuje server dlouho držet spojení aktivním. Metoda `POST` se používá při např. vyplnění uživatelem formuláře nebo nahrávání videa na server.
Oba útoky jsou těžko detekovatelné a vyžadují minimální kapacity připojení.

1.4.5 SSDP amplification

SSDP (Simple Service Discovery Protocol) je jedním z protokolů sady UPnP (Universal Plug and Play), který umožňuje sit'ovym zařízením podporujícím UPnP protokol, inzerovat své služby v síti. Protokol SSDP je využít v takových zařízeních jako jsou: domácí routery, audio systémy, tiskárny, IP kamery a další. Uživatel nemusí tyto zařízení konfigurovat, sada protokolů UPnP to dělá automaticky. Ostatní zařízení se můžou dotázat zprávou M-SEARCH na multikastovou adresu a port (239.255.255.250:1900) pro nalezení zařízení zájmu.

```
M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
MAN: "ssdp:discover"
MX: 1
ST: urn:dial-multiscreen-org:service:dial:1
USER-AGENT: Vivaldi/78.0.3904.99 Linux
```

Výše je uvedená zpráva M-SEARCH a její pole. Důležité pro nás je pole ST (search-target) kde se uvádí zařízení zájmu. V tomto případě je uveden specifický typ zařízení, ale existují dva standardní typy:

- `upnp:rootdevice`: hledá root zařízení,
- `ssdp:all`: hledá všechny UPnP zařízení a služby.

Největší amplifikace vzniká u druhého typu. Podvržením IP adresy tak nasměrujeme odpovědi od všech zařízení v síti na stanici obětí. Odpověď vypadá následovně:

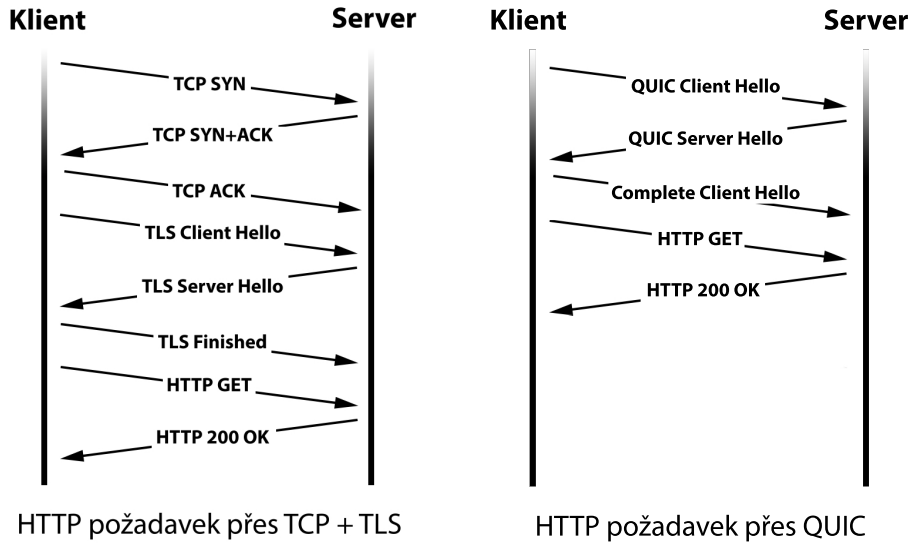
```
HTTP/1.1 200 OK
CACHE-CONTROL: max-age=100
DATE: Thu, 21 Nov 2019 11:37:45 GMT
EXT:
LOCATION: http://192.168.0.1:1900/igd.xml
SERVER: ipos/7.0 UPnP/1.0 TL-WR841N/9.0
ST: urn:schemas-upnp-org:device:InternetGatewayDevice:1
USN: uuid:<...>::urn:<...>:device:InternetGatewayDevice:1
```

Na jedinou zprávu M-SEARCH (cca 210 bajtů) od každého reflexního zařízení přijde 7-8 odpovědi, každá z odpovědí má zhruba 340 bajtů. Pole LOCATION obsahuje odkaz na popis služeb které zařízení podporuje. Volání těchto služeb probíhá přes SOAP (Simple Object Access Protocol).

Problémem SSDP protokolu je to, že není kontrolováno z jaké sítě přichází požadavek tzn. jakékoliv zařízení ve veřejné síti se může dotázat zprávou M-SEARCH. UPnP protokol neimplementuje žádnou autentizaci, tato bezpečností díra většinou ohrožuje routery které podporují UPnP. Největším rizikem může být přenastavení DNS serveru a přesměrování portů na routeru. To je možné zneužitím podporovaných služeb zařízení. V prvním případě útočník podvrhne DNS server tak, že při pokusu navštívit stránku xyz.com, klient bude přesměrován na útočníkův zfalšovaný web. Přesměrování portů dovolí útočníkovi navázat spojení s lokálním počítačem který se umisťuje za routerem (pokud je router veřejně přístupný). Nastavením politiky zákazu příchozího trafiku na UDP port 1900 je možné se zabezpečit proti SSDP amplifikaci. Avšak dnes hodně UPnP zařízení postavených na knihovně `libupnp` odpovídá z efemérních zdrojových portů, což dělá detekci útoku obtížnější. Skenováním celého internetu nejvíce odezev UPnP zařízení pocházelo z Číny [6].

1.4.6 QUIC attack

QUIC (Quick UDP Internet Connections) je protokol poprvé navržený firmou Google, který představuje zcela nový způsob přenosu informací na internetu. Hlavní rozdíl je ve využití UDP protokolu namísto obecně využívaného TCP protokolu. QUIC kombinuje rychlost UDP protokolu a spolehlivost TCP protokolu. Navázání zabezpečeného HTTPS spojení probíhá zasláním jedné až dvou zpráv (počet zpráv závisí na tom jestli klient navazuje spojení s novým nebo již známým serverem). Menší počet zasílaných zpráv je dosažen především nahrazením TCP a TLS protokolů. Požadavek na obdržení HTTP stránky se stal rychlejší, viz obrázek 1.5.
Standardizaci protokolu převzala skupina IETF (Internet Engineering Task Force), nicméně Google paralelně stále vylepšuje svoji implementaci protokolu který nese název gQUIC. Přestože implementace QUIC od IETF je odvozena od implementace gQUIC, kvůli značným změnám ze strany IETF, můžeme považovat QUIC jako separátní protokol [7].



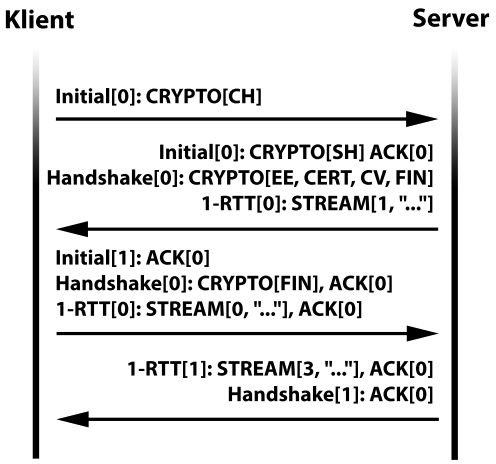
Obr. 1.5: Srovnání počtu zpráv QUIC oproti TCP+TLS na načtení HTTP stránky.

Kvůli stálým vylepšováním protokolu vznikají jeho různé verze. Aktuální verze je `draft-27` od IETF a `Q050` od Google. Servery s podporou QUIC se snaží implementovat hned několik verzi protokolu.

Typickým scénářem útoku na servery využívající QUIC je zaslání inicializační zprávy od klienta na vyžadování spojení. Klient ve své první zprávě posílá parametry pro QUIC spojení spolu se zprávou `ClientHello` (obdobně jako u TLS protokolu) která obsahuje parametry pro zabezpečené spojení. Parametry pro zabezpečené spojení jsou inkapsulovány do `CRYPTO` rámce QUIC protokolu.

Server odpovídá na požadavek klienta dvěma až třemi zprávami, které jsou obvykle dva až třikrát větší než inicializační zpráva klienta. Tím vzniká amplifikace. Konkrétněji server odpovídá inicializační (`Initial`) zprávou, která může v sobě obsahovat `Handshake` a `1-RTT data` zprávy, jinak jsou tyto zprávy posílané zvlášť. Zpráva `Initial` od serveru obsahuje `ServerHello` zprávu inkapsulovanou do `CRYPTO` rámce. Stejně tak do `CRYTPO` rámce je inkapsulována `Handshake` zpráva, která obsahuje certifikát serveru. Hned po odeslání těchto dvou zpráv server může poslat zašifrovaná data [8]. Detailnější znázornění posílaných zpráv viz obrázek 1.6.

Amplifikační útok není jediná možnost zneužití QUIC protokolu. Jelikož servery podporující QUIC protokol mají otevřený UDP port 443, je možné zneužití QUIC protokolu záplavovým útokem.



Obr. 1.6: Navázání nového spojení pomocí QUIC protokolu.

0-RTT útok

Technika 0-RTT (zero round-trip time) umožňuje uživateli poslat data ještě před navázáním spojení se serverem. Klient zašle 0-RTT paket serveru se kterým předtím již komunikoval a na základě dříve obdžených kryptografických dat. Po přijetí paketů server naváže zcela nové spojení s použitím nových kryptografických dat. Technika 0-RTT vznikla spolu s verzí TLS 1.3 protokolu a je zařazena do implementace QUIC protokolu. Ve výchozím nastavení je tato technika vypnuta na většině serverů.

Útočník ale nic ze zachyceného 0-RTT paketu nezjistí, jelikož je zašifrovaný. Problémem je, že daný paket může zduplikovat a poslat několikrát na stejný server. Například při zachycení 0-RTT paketu uživatele který posílá peníze a jeho opakované odeslání útočníkem, může potenciálně provést transakci vůči uživateli několikrát [9].

2 Zátěžové testováni

Zátěžové testování je bezpodmínečné pro vytvoření kvalitní služby. Každá taková služba má důležitou charakteristiku, jako např. hranice zatížení. Při překročení této hranice nastává výpadek systému, který se může projevovat různými způsoby (nejčastěji nedostupnosti služby). Ne vždy příčinou nedostupnosti je zvýšený zájem uživatelů o službu v určitý časový okamžik. Distribuovaný útok je vhodný při posuzování chování systému ve stresové situaci. Zátěžové testování, kromě webové služby samotné, pokrývá značnou část testování síťové infrastruktury, zejména velký důraz je kladen na hardwarové firewally. Pro interpretaci výsledků zátěžového testu se používají níže uvedené metriky [10].

Průměrný čas odezvy – průměrný čas od okamžiku zaslání žádosti klientem do obdržení odpovědi od serveru, vypočítaný pro všechny transakce (žádost/odpověď) v určitý časový interval.

Maximální doba odezvy – nejdelší doba provedení jedné transakce. Užitečná metrika pro odhalení potenciálně problematického místa systému.

Chybovost – procent selhaných žádostí oproti celkovému počtu vygenerovaných žádosti. Udává se v procentech [%].

Počet souběžných uživatelů – vyjadřuje počet aktivních virtuálních uživatelů v určitý okamžik.

Žádosti za vteřinu – počet žádostí na stažení HTML stránky, XML dokumentu, JavaScript knihoven, obrázků atd.

Propustnost – využitá šířka přenosového pásma. Ukazuje aktuální rychlost datového toku u síťového rozhraní v určitý časový okamžik.

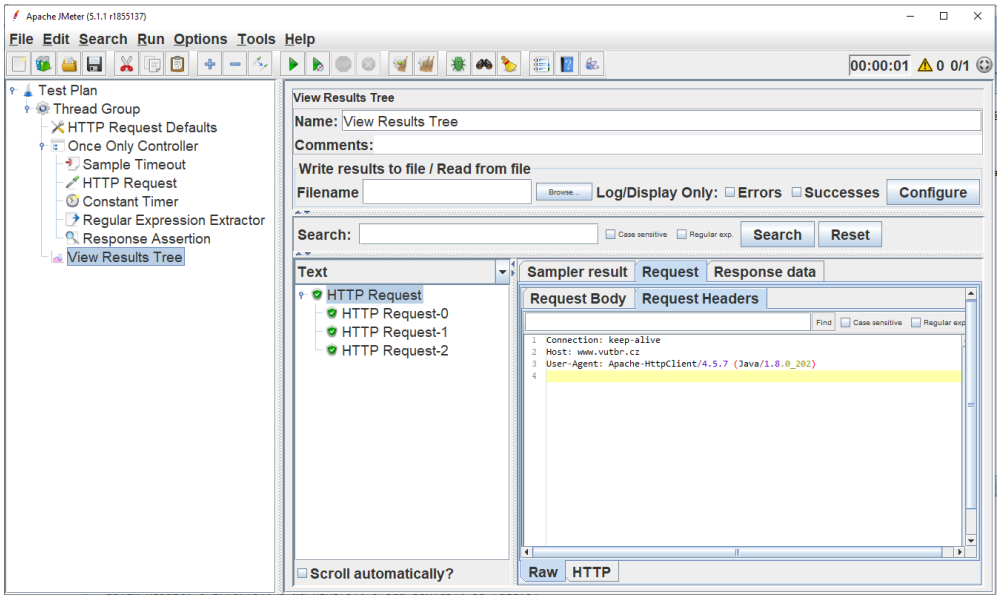
2.1 Apache JMeter

Apache JMeter je nástroj určený pro zátěžové a výkonnostní testování aplikací. Umožňuje jak manuální vytváření testovacích scénářů skládáním jednotlivých komponent, tak i jejich dynamické vytváření zaznamenáváním činnosti uživatele ve webovém prohlížeči. Je to robustní a hlavně flexibilní nastoj, který dokáže poměrně přesně nasimulovat chování uživatele. Umožňuje otestovat velkou škálu aplikací, servery, protokoly jako HTTP(S), FTP, TCP, Webové služby atd. Disponuje grafickým uživatelským rozhraním a je kompatibilní se všemi operačními systémy. Aplikace je napsaná čistě v jazyce Java [11].

2.1.1 Základní elementy

- **Test Plan** – hlavní komponenta která obsahuje všechny testovací kroky. Slouží pro vkládání ostatních elementů. Představuje individuální scénář.
- **Thread Group** – definuje počet virtuálních uživatelů. Jedná se o počáteční bod každého testovacího plánu, všechny vzorky (dále jen sampler) a kontroléry se musí rozmísťovat pod tímto elementem.
- **Sampler** – základní element simulující požadavek na server přes určitý protokol.
- **Logic Controller** – definuje logiku a stanovuje pořadí vykonání sampleru.
- **Config Element** – nastavuje proměnné a defaultní hodnoty pro samplery.
- **Listener** – ukládá výsledky provedení testu a zobrazuje je ve formátu tabulky, grafu, logu nebo stromového diagramu.
- **Timer** – nastavuje prodlevu mezi provedením vybraných sampleru. Bez použití timeru Jmeter odesílá požadavky s největší možnou rychlostí, což může zahltit server.
- **Pre processors** – umožňuje provést určité operace před provedením sampleru. Např. uložit hodnotu do proměnné, získat data z databáze atd.
- **Post processors** – provádí určité operace po provedení sampleru. Nejvíce se používá pro získání hodnot z odpovědi serveru, např. získání identifikátoru relace.
- **Assertions** – srovnává skutečnou a očekávanou odpověď od serveru, v případě neshody hodnot vyhodnotí požadavek za neúspěšný.

Testovací plán který využívá všechny výše zmíněné elementy je vidět na obrázku 2.1



Obr. 2.1: Scénář s využitím všech elementů aplikace JMeter.

Pořadí provedení výše uvedených elementů resp. skupin elementů je následující:

1. Config Element,
2. Pre processors,
3. Timer,
4. Sampler,
5. Post processors,
6. Assertions,
7. Listener.

2.1.2 Rozšíření aplikace JMeter

Plugin Manager

Nejjednodušší možnost rozšíření aplikace JMeter je pomocí již vytvořených modulu. Jelikož aplikace je na bázi open source a šiřitelná pod licencí Apache Licence v2.0, každý může stáhnout zdrojový kód a přidat svoji funkcionalitu. `Plugin Manager` je rozšíření pomocí kterého se snadněji instalují a udržují nové moduly. Uživatel má možnost vybrat z více než 70 přídavných modulů. `Plugin Manager` není ve výchozím nastavení programu JMeter nainstalován a uživatel si jej musí stáhnout samostatně.

Vlastní modul

Další možnosti rozšíření aplikace JMeter je vytvoření vlastního modulu. Při tvorbě aplikace vývojáři kladli důraz na její snadnou rozšiřitelnost. Při vyvíjení vlastního modulu je třeba dodržovat strukturu projektu, kterou ve velké míře definuje Apache Ant – nástroj pro sestavování softwarových aplikací.
Existují celkem dva způsoby rozšíření: s použitím nového TestBean frameworku a bez jeho použití. V obou případech uživatel plně implementuje logiku modulů samostatně, rozdílem je přístup v napsání grafického uživatelského rozhraní (dále GUI) a propojení s třídou implementující logiku. Tedy v prvním způsobu má uživatel sadu předdefinovaných grafických elementů a s dodržením názvů souborů, propojení logiky a grafického uživatelského rozhraní probíhá pomocí TestBean frameworku. U druhého způsobu je propojení definováno uživatelem, výhodou je větší flexibilita při vytváření GUI s využitím Swing knihovny.

2.2 Nástroj Trafgen

Trafgen je rychlý generátor síťového provozu a je součástí balíčku `netsniff-ng`. Je vyvinutý výhradně pro operační systémy založené na Linux architektuře. Nástroj je distribuován pod licencí GNU General Public License v2.0, jeho autorem je Daniel

Borkmann. Trafgen je určený především pro ladění a testování výkonnosti sítě. Pro sestavení paketu se využívá vlastní nízkoúrovňový konfigurační jazyk , který není limitován na určitý protokol. Díky tomu se uplatní také ve *fuzz* testování, což je technika pro odhalení potenciálních chyb v systému metodou odesílání chybných, neočekávaných nebo náhodných dat. Sestavený paket je možné vidět ve výpise 2.1. Jedinou limitací nástroje je neschopnost vytvořit plnohodnotnou relaci. Trafgen je vícevláknový nástroj a implicitně spouští maximální počet procesů podle toho kolik je dostupných CPU. Vysoké výkonnosti je dosaženo využitím *zero-copy* mechanismu, kdy při odesílání a přijetí pakety nemusí byt kopírované z prostoru jádra do uživatelského prostoru a opačně [12].

Výpis 2.1: Ukázka konfiguračního souboru Trafgen.

```
{
  # --- ethernet hlavička ---
  eth(da=00:50:55:ea:ee:26, sa=00:0c:29:fa:60:d7),
  # --- ip hlavička ---
  ipv4(id=drnd(), ttl=64, sa=10.1.2.1, da=45.88.126.9),
  # --- tcp hlavička ---
  tcp(sport=drnd(), dport=80, seq=drnd(), syn, win=16),
}
```

2.3 Testování síťové infrastruktury

Testování síťové infrastruktury probíhalo pomocí nástroje vyvíjeného v rámci projektu „Zátěžový tester ICT“ v laboratorní místnosti FEKT VUT Brno. Zátěžový tester je rozšířením aplikací Apache JMeter a je propojený s knihovnou Trafgen. Nástroj implementuje v sobě různé druhy DoS útoků určené pro generování zátěže. Obsahuje taky realizaci webového serveru, sloužící pro zpracování požadavků a další podpůrné moduly pro sledování průběhu komunikace.

Útoky typu DoS spotřebovávají velké množství systémových prostředků a při jejich vytížení se projevuje nestabilita fungování systému. Hardwarová soustava zátěžového testeru musí být dostatečně výkonná aby byla docílena co největší přesnost měření.

Bylo použito následující hardwarové vybavení:

- Procesor Intel Xeon E5-2650 v4 o frekvenci 2.2 GHz, 12 jader,
- Základní deska ASUS X99-E WS,
- Operační paměť 128 GB 2133 MHz,
- Grafická karta NVIDIA GeForce GT 730,
- Síťová karta Intel X540-AT2, 2x 10 GbE.

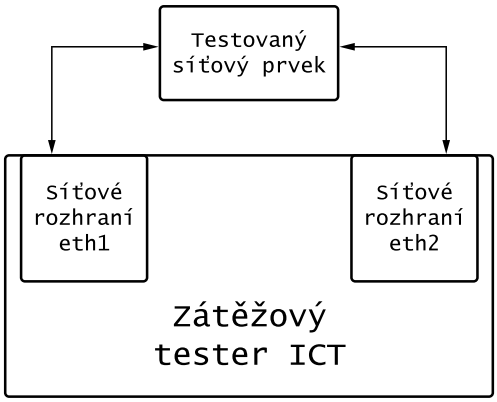
Softwarovou část zátěžového testeru tvoří:

- Operační systém CentOS 7.3.1611, JMeter 4.0, trafgen 0.6.3, Apache Server 2.4.6, Java OpenJDK 1.8.0_101.

Dále jsou uvedené jednotlivé scénáře, popisující testování pomocí modulů realizovaných v zátěžovém testeru.

2.3.1 Testování mezilehlého síťového prvku

Komunikace mezi klientem a serverem probíhala přes mezilehlý síťový prvek, jenž byl testován. Pomocí zásuvného modulu Network Analyzer byl sledován provoz na rozhraních `eth1` a `eth2` výše uvedené síťové karty. Zapojení sítě je zobrazeno na obrázku 2.2. Jako mezilehlý síťový prvek byl použit router DrayTek Vigor2920Vn s IP adresou 192.168.1.1. Implementované útoky DoS (jejich samplery) bylo možné



Obr. 2.2: Zapojení zátěžového testeru ICT a síťového prvku.

použít jen v rámci dvou Thread Groups. Kvůli nestabilnímu a pomalému výkonu modulu DDoS Stairs Thread Group, byl v testovacím plánu použit modul DDoS Simple Thread Group, při kterém je generovaná konstantní úroveň zátěže. Za zpracování požadavku odpovídal Apache Server, jehož konfigurace probíhala přes modul Server Emulator. Apache Server naslouchá na rozhraní `eth1` a má IP adresu 192.168.1.11. Zátěžový tester odesílá pakety přes rozhraní `eth2` a má IP adresu 192.168.1.12. Měření probíhalo s využitím modulu UDP flood s předdefinovanou délkou rámce 60 bajtů.

Cílem testovacího případu je měření zátěže, při které router přestane obsluhovat požadavky a dojde k jeho restartování. Zvolená konstantní délka trvání testu je 20 vteřin a v každém dalším cyklu měření se zvyšuje počet paketů za vteřinu (pps). Výsledek testu také zahrnuje informace o vytížení CPU a RAM, viz obrázek 2.4.

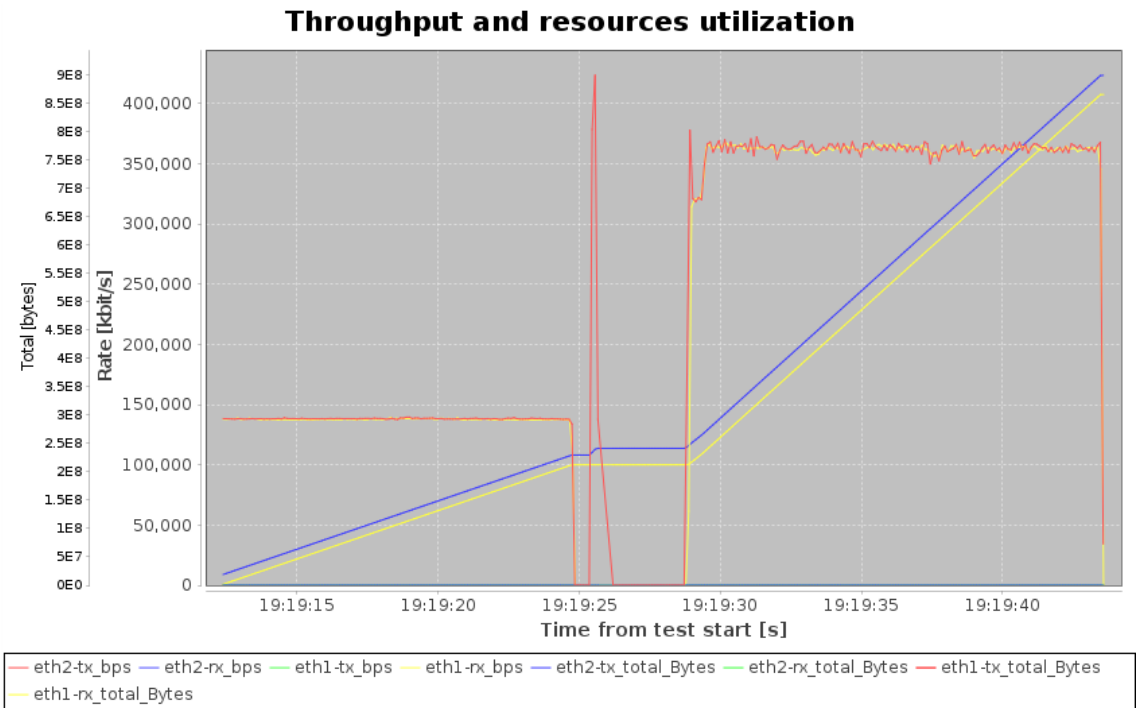
Statistiky byly získané pomocí modulu Network Analyzer, jenž sleduje odchozí tok z rozhraní `eht2` a příchozí tok na rozhraní `eth1`. Výsledek testu je zobrazen pro zátěž 1 milion UDP paketů za vteřinu. V průběhu celého testu nebylo webové rozhraní směrovače DrayTek dostupné.
Podle obrázku 2.3 je vidět, že po dobu 10 vteřin maximální propustnost routeru byla na úrovni 138 Mbit/s. Při této úrovni ustálené zátěže nastal výpadek routeru a proběhl jeho automatický restart. Stejného výsledku měření bylo dosaženo odesláním 100 tisíc UDP paketů.
V rámci tohoto testu byl paralelně testován modul SYN flood, výsledek testu byl téměř identicky. Hlubším zkoumáním fungování modulu SYN flood bylo zjištěno, že při odeslání paketu s příznakem SYN na server, nebyla obdržena odpověď SYN+ACK ze strany serveru. Následně bylo zkoumáno jestli je problém způsoben ze strany serveru nebo nesprávnou funkčností modulu. V programu Wireshark byl zachycen síťový provoz na rozhraní `eht2` a zjistilo se, že při odeslání požadavku na server na načtení stránky přes webový prohlížeč, byla relace navázána a požadovaná stránka se zobrazila. Nefunkčnost se tedy projevuje na straně modulu. Pravděpodobně generované pakety přes modul SYN flood přichází na server poškozené, čímž je vysvětleno nepřijetí odpovědi SYN+ACK.

Modul Trafgen Custom Config

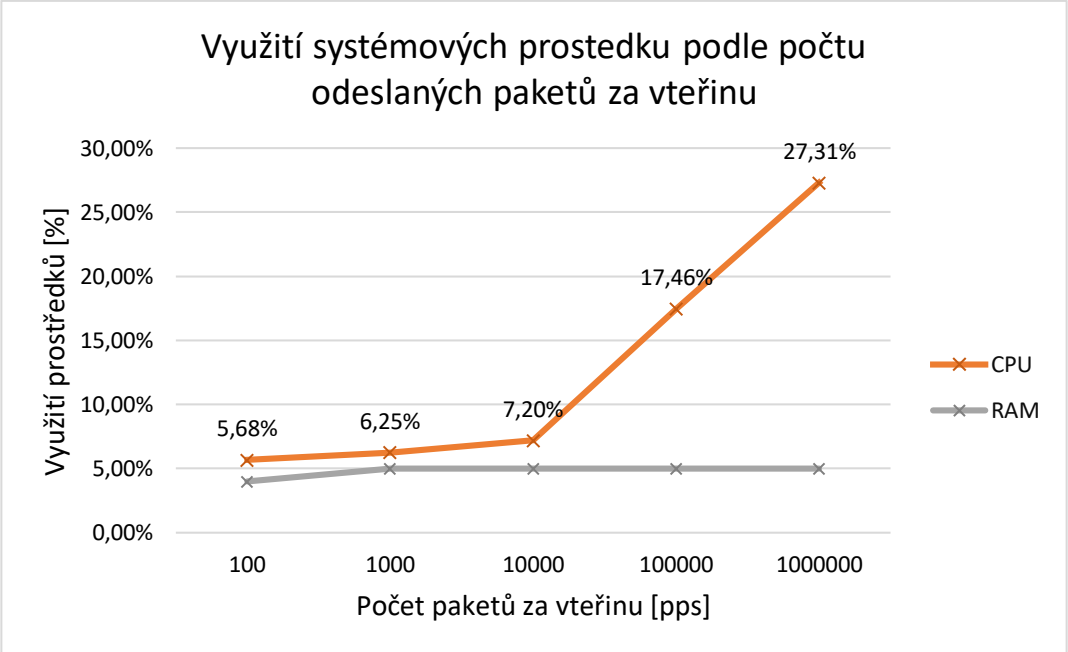
Modul slouží pro nastavení vlastního konfiguračního souboru nástroje Trafgen. Cílem jeho použití byla snaha provést stejný test za větší velikosti UDP paketů, jelikož modul UDP flood toto nastavení neimplementuje. Modul se nepodařilo spustit a v logu se objevila následující chybová hláška:
Uncaught Exception java.lang.NullPointerException.

2.3.2 Testování modulu Network Emulator

Modul je vložen na začátku každého testovacího plánu a je implicitně ve vypnutém stavu. Network Emulátor je primárně určen pro emulaci přenosových parametrů mezi dvěma uzly existující sítě. Zapnutím daného modulu vybereme síťová rozhraní, přenosové parametry kterých chceme ovlivnit. Modul umožňuje emulovat propustnost, zpoždění, ztrátovost, záměnu pořadí, duplikaci a chybovost paketového přenosu. Uvedené parametry jsou emulované vždy pro odchozí datový tok daného rozhraní.
Pro funkčnost modulu je potřeba vytvořit nový testovací plán v programu JMeter. Nastavená ztrátovost na odchozí datový tok rozhraní `eth2` byla 75 %, spuštěním testovacího plánu aplikujeme danou konfiguraci. Pro ověření funkčnosti byl odeslán příkaz *ping* na adresu routeru, výsledek je zobrazen na obrázku 2.5. Ověřena byla

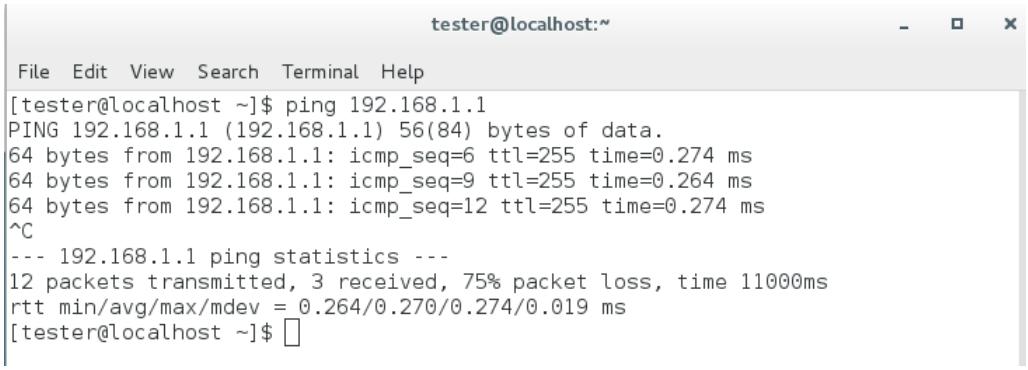


Obr. 2.3: Graf průběhu přijatých a odeslaných rámců útoku UDP flood.



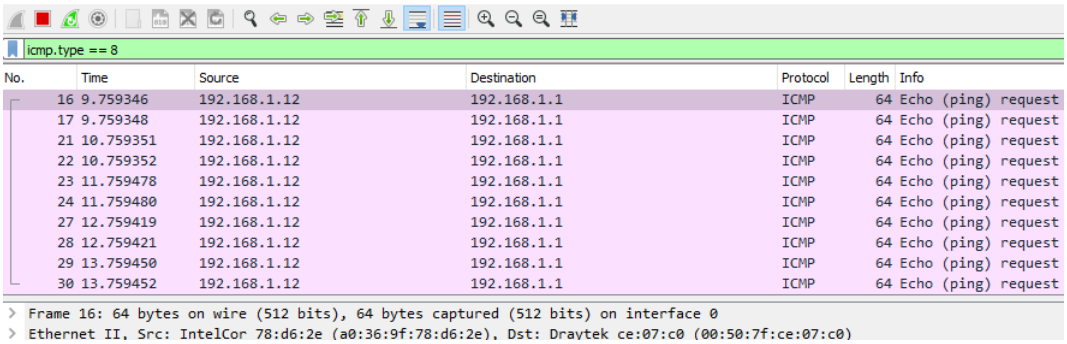
Obr. 2.4: Vytížení CPU a RAM paměti během testu UDP flood.

taky funkčnost dalších parametrů modulu, všechna nastavení emulace bez problémů fungovala v rámci nativních komponent programu JMeter.



Obr. 2.5: Nastavená ztrátovost modulu Network Emulátor.

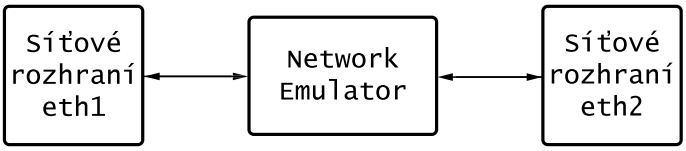
Dále bylo ověřeno fungování modulu spolu s implementovanými moduly pro DDoS útoky. Žadné nastavení nezpůsobilo změnu v datovém toku. S pomocí funkcí *Port Mirroring* routeru DrayTek, byl zachycen veškerý datový tok který přišel na router. Princip této funkce je následující: všechny pakety které dorazí na určitý port routeru, budou kopírované a přesměrují se na jiný port. K tomuto portu byl připojen další počítač a monitoroval síťový provoz programem Wireshark. Zátěžovým testerem bylo odesláno 10 ICMP paketů při nastavení ztrátovosti 75 %, na obrázku 2.6 je vidět, že všechny pakety byly doručené na router a žádný se nezahodil. Zachytávání paketů tímto způsobem ukazuje, že skutečně žádná změna v datovém toku nebyla provedena. Příčinou této nefunkčnosti je pravděpodobně neschopnost Network Emulatoru zachytit paket, generovaný nástrojem Trafgen, z jádra systému.



Obr. 2.6: Zachycené pakety pomocí funkcí Port Mirroring.

Scénář s využitím nativních komponent programu JMeter

Komunikace mezi klientem a serverem probíhala bez využití mezilehlého síťového prvku, zapojení je zobrazeno na obrázku 2.7. Spouštěním testovacího plánu, který je vidět na obrázku 2.8, zahájíme relaci. V rámci jedné relace proběhne nejdřív třícestný handshake a provede se tím synchronizace se serverem. Následně proběhne samotné odeslání HTTP požadavků na načtení stránky a nakonec rozvázání spojení odesláním TCP paketů s příznakem FIN. Nastavený počet uživatelů (relací) je 10 tisíc, které se za dobu 20 vteřin musí připojit na server. Ovlivněný parametr emulace je ztrátovost paketů. Výsledky testů jsou uvedené v tabulce 2.1, výsledná hodnota je průměrem z pěti měření pro každý testovací případ. Sledované metriky jsou: průměrná doba odezvy, maximální doba odezvy, počet žádostí za vteřinu, celkový čas testu. Nastavení ztrátovosti bylo aplikováno na síťové rozhraní `eht2`, z důvodu skrytí rozhraní `eth1` na kterém naslouchal server.



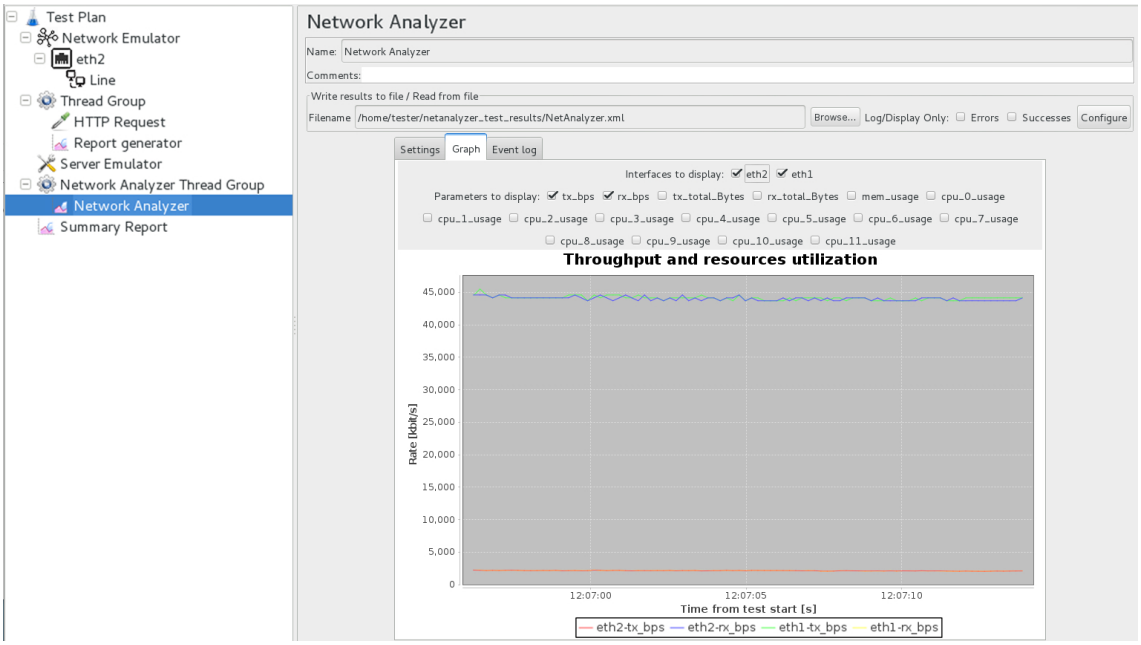
Obr. 2.7: Zapojení modulu Network Emulator.

Tab. 2.1: Výsledky měření s nastavením ztrátovosti modulu Network Emulator.

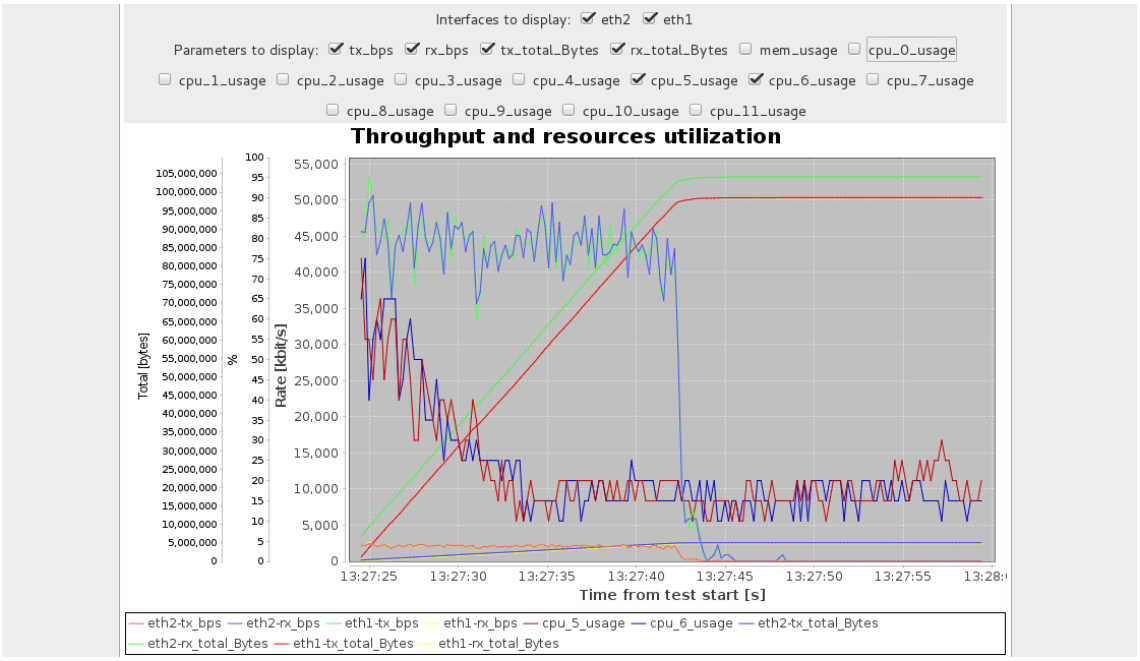
Ztrátovost	0 %	5 %	10 %	15 %	30 %
Průměr[ms]	0	66	152	244	851
Max[s]	0,11	7,02	15,04	31,27	83,21
Žádostí/s	485,2	435,8	364,8	279	101,8
Délka testu[s]	20	22,5	26,5	35	168

V případě ztracení paketu musí proběhnout jeho opětovné zaslání, což v důsledku zapříčiní delší dobu relací. Nastavená ztrátovost je pouze demonstrační a slouží pro ověření funkčnosti zásuvných modulů zátěžového testeru. Maximální přípustná hodnota ztrátovosti paketů v reálu činí 2 %, větší hodnota ve značné míře ovlivní uživatelskou zkušenost a službu považujeme za nekvalitní. Výsledek testu je také znázorněn ve formě grafu, generovaného modulem Network Analyzer. Na obrázku 2.8 je zobrazen průběh testu za ideálních podmínek, obsahuje hodnoty přijatých a odeslaných dat pro obě rozhraní. Testovací případ s nastavenou ztrátovostí paketů 15 % je

vidět na obrázku 2.9, navíc je přidána hodnota využití CPU (linie uprostřed). Linie nahoře zaznamenávají vytíženost linky při odeslání dat ze síťového rozhraní `eth1` a přijetí dat na síťovém rozhraní `eth2`. V případě nastavené ztrátovostí je viditelný pokles linií, který je vysvětlený opětovným odesláním zahozených paketů.



Obr. 2.8: Průběh testu bez emulaci přenosových parametrů.



Obr. 2.9: Průběh testu s emulací ztrátovosti paketového přenosu 15 %.

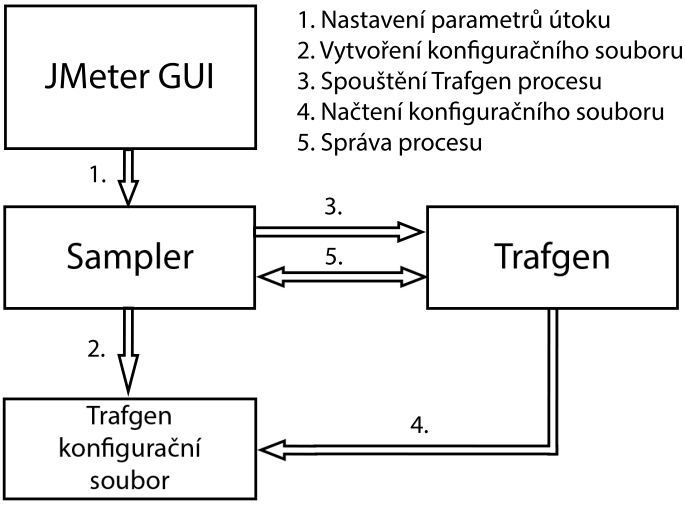
3 Rozšíření nástroje JMeter

Propojení nástroje JMeter a knihovny Trafgen

Implementace modulu pro zátěžové testování je založena na externí knihovně Trafgen (kapitola 2.2). Trafgen slouží jako generátor síťového provozu a je ideálním řešením při využití v DoS útocích, které nepotřebují navázání spojení.

Konfigurace útoku se provádí přes grafické uživatelské rozhraní JMeteru. Uživatel si nastaví parametry útoku, které se uloží do konfiguračního souboru. Spouštěním sampleru v JMeteru se na pozadí provede vytvoření nového Trafgen procesu, který načte konfigurační soubor a provede požadovanou operaci. Proces se ukončí buď termínovaně uživatelem z kontextu aplikace JMeter nebo po provedení operace.

Grafické znázornění propojení aplikace JMeter a Trafgenu je zobrazeno na obrázku 3.1



Obr. 3.1: Propojení aplikace JMeter a nastoju Trafgen.

Struktura Apache JMeter

Struktura aplikace Apache JMeter je tříděna logicky, aby byl vývoj nových komponent přehlednější. Hlavní složky se kterými uživatel pracuje v rámci vývoje modulu jsou:

- **bin** – obsahuje soubory pro spouštění aplikace JMeter.
- **build** – obsahuje .class soubory vygenerované po korektní kompilaci zdrojových kódů.

- **lib** – umisťuje zkompilované .jar soubory se kterými napřímo pracuje aplikace JMeter.
- **src** – obsahuje zdrojové kódy.

Složka **src** je jádrem programu a obsahuje v sobě tyto podsložky:

components	realizace modulů které nesouvisí s protokolem, tj. timery, kontroléry atd.
core	hlavní logika JMeteru na které je vybudovaná aplikace.
example	obsahuje ukázkové samplery na vytvoření modulu.
function	obsahuje funkce které jsou využité napříč všemi moduly.
jorphan	realizace podpůrných funkcí.
junit	realizace junit modulů sloužící k unit testům.
protocol	obsahuje realizaci protokolu které JMeter podporuje.

Struktura přídavného modulu JMeter

Uživatel si má možnost vybrat z dvou variant jak bude vypadat struktura přídavného modulu. Ukázkové moduly jsou umístěné ve složce `src/examples`. Od verze Apache JMeter 2.0 je podporována možnost využití TestBean frameworku pro přidávání nových komponent aplikace. Struktura modulu s využitím TestBean frameworku bude popsána dále. Klasická varianta modulu obsahuje dvě základní třídy: `Sampler.java` a `SamplerGui.java`.

Sampler.java

Definuje základní logiku modulu. Modul musí patřit do některé ze skupin základních elementů, je popsané v 2.1.1, proto potřebuje zdědit příslušnou třídu. Například třída `AbstractSampler` je zděděna v případě kdy chceme vytvořit element sampler, nebo třída `ConfigTestElement` v případě elementu konfigurace. Dále každý modul musí mít implementované rozhraní `TestBean`.

SamplerBeanInfo.java

Obsahuje definici a popis grafických elementů, které se vyskytují v GUI modulu. Definice je přiřazení elementu ke konkrétnímu tvaru, například jestli to bude textové pole, nebo rozbalovací seznam. Popis říká jaká je počáteční hodnota elementu, jestli element je povinný atd. Název této třídy musí být `SamplerName + BeanInfo` a musí dědit ze třídy `BeanInfoSupport`.

SamplerResources.properties

Soubor s textovými řetězci pro každý grafický element nebo skupinu elementů. Formát zápisu je `klíč=hodnota`. V podstatě se jedná o definici názvu elementu a definici textu nápovědy při umístění kurzoru na element. Název tohoto souboru musí být ve formátu `SamplerName + Resources`.

Dodržování názvů tříd je hlavním aspektem při vývoji modulu. Každá použitá proměnná musí mít vlastní metodu `get` a `set`, jinak se grafické rozhraní modulu nebude zobrazovat správně.

3.1 Konfigurace vývojového prostředí

Vývoj modulu DoS útoku probíhal na virtuálním počítači s operačním systémem CentOS 8, který běžel na hostitelském sytému s Windows 10 pomocí programu VMware Workstation 15.

Instalace JMeter

Na oficiálních stránkách Apache JMeter má uživatel možnost vybrat si ze dvou variant: `source` anebo `binary` vydání programu. Rozdíl `source` oproti `binary` je ten, že první obsahuje zdrojové kódy a před samotným spouštěním programu je nutné zdrojové kódy zkompilovat u sebe na počítači. Vydání `binary` je předkompilované na jiném počítači a distribuje se bez dodání zdrojových kódů. Pro účely rozšíření aplikace JMeter musíme stáhnout `source` verzi. Aktuální stažená verze v době vývoje je Apache JMeter 5.1.1.
Vývojové prostředí Eclipse Oxygen je použito pro ulehčení a přehlednost při vývoji modulu. Apache Ant je použit pro sestavování aplikace tzn. její správnou kompilaci. Ant definuje tzv. `targets` které najdeme v souboru `build.xml` v kořenové složce projektu JMeter. Konfigurace projektu se provede spouštěním následujících `targets`, viz výpis 3.1, které provedeme z příkazové řádky v kořenové složce projektu.

Výpis 3.1: Počáteční konfigurace projektu Apache JMeter.

```
ant download_jars #stáhne potřebné .jar závislosti
ant install       #sestavení aplikace
```

Instalace Trafgen

Trafgen je součástí balíčku `netsniff-ng`, který v sobě obsahuje sadu dalších síťových nástrojů. Podrobná instalace Trafgen viz výpis 3.2

Výpis 3.2: Podrobná instalace nástroju Trafgen.

```
dnf group install "Development␣Tools"
git clone git://github.com/netsniff-ng/netsniff-ng.git
cd netsniff-ng
make trafgen
```

Provedením výše uvedených příkazů se vygeneruje spustitelný soubor ve složce `/netsniff-ng/trafgen`. Zkopírováním spustitelného souboru do složky `/usr/sbin/`, umožníme uživateli spouštět Trafgen mimo jeho kořenový adresář. Instalovaná verze Trafgen je 0.6.6.

Generování konfiguračního souboru Trafgen

Generování konfiguračního souboru neboli paketu v nástroji Trafgen je jednodušší s využitím hlavičkových funkcí protokolů. Syntaxe těchto funkcí viz výpis 3.3.

Výpis 3.3: Obecný tvar hlavičkových funkcí.

```
<protokol>(<parametr>=<hodnota>,<parametr2>=<hodnota2>,...)
```

Počet hlavičkových funkcí je omezen ale přesto se dá sestavit libovolný konfigurační paket. Možností je například odchytit paket který nás zajímá programem Wireshark, uložit do `.pcap` souboru a pomocí příkazu viz výpis 3.4, převést `.pcap` soubor do Trafgen `.cfg` konfiguračního souboru.

Výpis 3.4: Převádění .pcap souboru do formátu Trafgen .cfg souboru.

```
netsniff-ng --in foobar.pcap --out foobar.cfg
```

Provedením výše uvedeného příkazu se vytvoří konfigurační soubor v hexadecimálním formátu. Obsah konfiguračního souboru Trafgen můžeme vyjádřit i v dalších formátech viz tabulka 3.1.

Tab. 3.1: Podporované formáty konfiguračního souboru Trafgen.

hexadecimálně	0xf9, x73
dekadicky	31
binárně	0b11110000, b11110000
osmičková soustava	011
znaková sada	'a'
řetězcový	"hello world"
shellkod	"\x31\xdb\x17\x99\xcd\x80\x31\xc9"

Kombinací hlavičkových funkcí a formátovaného obsahu můžeme sestavit libovolný konfigurační paket.

Přidání vlastního Ant target pro přídavné moduly JMeter

Moduly aplikace JMeter jsou logicky řazeny po jednotlivých složkách jak je popsáno na začátku kapitoly 3. Kompilace modulu a přidání vygenerovaných `.jar` souborů se provádí spuštěním Ant `target` – `ant install`, který provede kompilaci celého projektu. Vytvořením vlastního Ant `target`, který bude zodpovědný za kompilaci jenom určitého kódu, eliminujeme nutnost kompilace celého projektu po každé provedené změně. Vytvořením vlastního `target` zachováme strukturu projektu, urychlíme proces tvorby a testování modulu a získáme samostatný `.jar` soubor pro DoS moduly.

Výpis 3.5: Ukázka Ant target pro kompilaci a instalaci přídavných modulů.

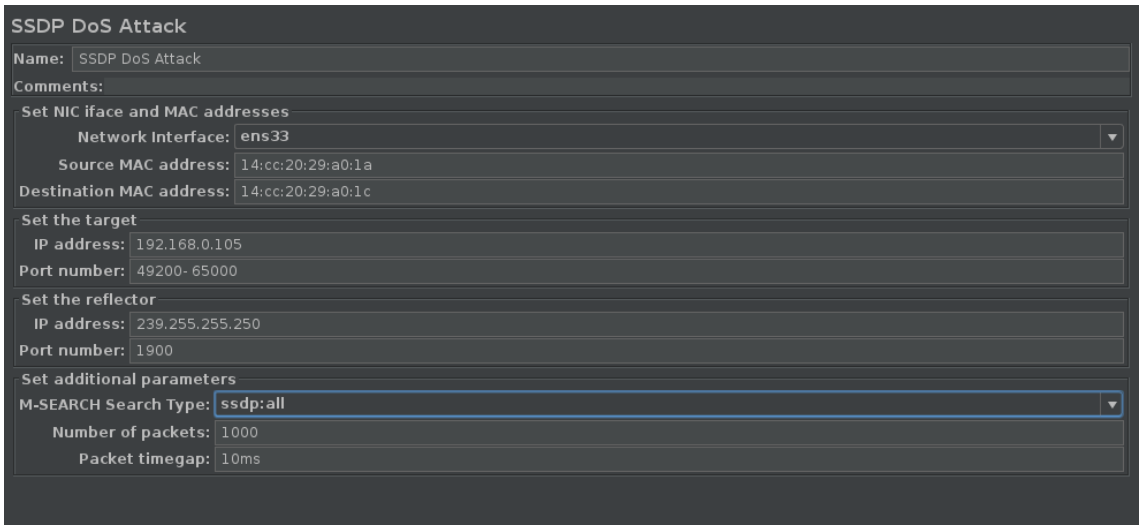
```
<target name="compile-custom" ....>
  <mkdir dir="${build.custom}"/>
  <javac srcdir="${src.custom}" destdir="${build.custom}" ....>
      <include name="**/*.java"/>
      <classpath>
        ....
      </classpath>
  </javac>
  <native2ascii src="${src.custom}" dest="${build.custom}" includes="
      **/*.properties" encoding="UTF-8"/>
</target>

<target name="install-custom" depends="compile-custom" ....>
  <jar jarfile="${dest.jar}/ApacheJMeter_custom.jar" ....>
       ....
      <fileset dir="${build.custom}" includes="**/*.class" />
      <fileset dir="${build.custom}" includes="**/*.properties" />
      <fileset dir="${build.custom}" includes="**/*.xml" />
  </jar>
</target>
```

Výpis 3.5 obsahuje definici `compile-custom` a `install-custom` Ant `targets`. Uvedené části kódu doplníme do souboru `build.xml`. Soubor obsahuje podobné `targets` pro ukázkové moduly JMeteru. Vynechané části výpisu doplníme z `targets` ukázkových modulů, které jsou společné.

3.2 Modul SSDP DoS attack

Vytvoření modulu SSDP útoku začíná sestavením kostry grafického rozhraní a třídy odpovídající za logiku modulu. Jako základ modulu je vhodné použít části kódu ukázkových modulů. Grafické uživatelské rozhraní modulu je definováno ve třídě `SsdpBeanInfo.java` a je zobrazeno na obrázku 3.2. Grafické prvky jsou rozdělené do jednotlivých skupin tak aby nastavení útoku bylo intuitivní.
Tento modul je rozdělen do čtyř skupin (shora dolů):

- Nastavení síťové karty a MAC adres – uživatel definuje síťové rozhraní ze kterého budou posílané pakety a také zdrojovou a cílovou MAC adresu. Většinou platí, že cílová MAC adresa je adresou následujícího zařízení zpracovávajícího daný paket, např. adresa routeru ke kterému je připojen počítač odesílající pakety.
- Nastavení cíle útoku – definuje se IP adresa a port zařízení na které chceme odpovědi z reflektoru směřovat.
- Nastavení reflektoru – definuje se IP adresa a port reflexního zařízení nebo multikastová IP adresa 239.255.255.250 pokud chceme posílat SSDP žádosti uvnitř sítí.
- Nastavení dalších parametrů – udávají se parametry specifické pro daný typ útoku a také parametry pro Trafgen.



Obr. 3.2: Grafické uživatelské rozhraní modulu SSDP DoS attack.

Grafické rozhraní modulu JMeter

Vytvoření elementů grafického rozhraní je poměrně přehledné. Třída odpovídající za definici grafických elementů je `PropertyDescriptor`, která je součástí balíčku

`java.beans`. Každé vstupní pole grafického rozhraní se navazuje na proměnnou určitého typu která je definována ve třídě `Ssdp.java`. Nejdříve musíme převzat pole metodou `property()` argumentem které zadáme proměnnou navazující na toto pole. Pomocí metody `setValue()` nastavíme vlastností pole, jako např. počáteční hodnota, povinnost pole atd.

Výpis 3.6: Ukázka definicí grafického prvku s využitím TestBean frameworku.

```
PropertyDescriptor p;
p = property("dstAddr");
p.setValue(NOT_UNDEFINED, Boolean.TRUE);
p.setValue(DEFAULT, "239.255.255.250");
p.setValue(NOT_EXPRESSION, Boolean.TRUE);
```

Výpis 3.6 obsahuje definici textového pole které nemůže mít nulovou hodnotu a defaultní hodnota kterého je **239.255.255.250**. Rozdělení elementů do logických skupin je realizováno pomocí metody `createPropertyGroup()` prvním argumentem které žádáme libovolný název identifikující skupinu, druhým argumentem které je pole typu `String` s požadovanými elementy.

Výpis 3.7: Přiřaděni grafických prvků do skupiny s využitím TestBean frameworku.

```
createPropertyGroup("destination_data", new String[]
  { "dstAddr", "dstPort" });
```

Výpis 3.7 znázorňuje seskupení elementů grafického rozhraní, který podle obrázku 3.2 odpovídá skupině pro nastavení reflexního zařízení (třetí shora).
Názvy grafických elementů a skupin které vidí uživatel jsou definovány v souboru `SsdpResources.properties`, kde je také uveden i název přídavného modulu který uživatel vidí v listě modulů.

Výpis 3.8: Nastavení názvů grafických prvků s využitím TestBean frameworku.

```
displayName=SSDP DoS Attack
destination_data.displayName=Set the reflector
dstPort.displayName=Port number
dstPort.shortDescription=Set the UPnP device port number
dstAddr.displayName=IP address
dstAddr.shortDescription=Set the IP address of a reflector
```

Vlastnost `displayName` nastavuje název elementu, když vlastnost `shortDescription` nastavuje text nápovědy při umístění kurzoru na element. Ve výpisu 3.7 jsou zmíněny elementy `dstAddr` a `dstPort` které odpovídají proměnným definovaným ve třídě `Ssdp.java` a které jsou popsané ve třídě `SsdpBeanInfo.java`.

Nastavení cíle útoku je možné zadáním určité IP adresy a portu nebo má uživatel možnost zvolit rozsah IP adres a portů na které budou směrované odpovědi z reflexního zařízení.

Spouštění a správa procesu modulu JMeter

Jelikož spouštění procesu Trafgen vyžaduje práva privilegiovaného uživatele, nejdříve si musíme ujasnit jak bude tento proces zavolán v kontextu modulu JMeter. První možností je spouštění aplikace JMeter s právy `root`, což se považuje za špatnou praxi. Aplikace spouštěná s `root` právy má možnost dělat jakékoliv úpravy v systému, proto existuje riziko poškození operačního systému. Aby nebylo nutné při spouštění testovacího plánu pokaždé zadávat heslo superuživatele, umožníme uživateli spouštět procesy Trafgen s `root` pravý bez udání hesla.

Výpis 3.9: Eliminace autentizace při spouštění Trafgen procesu.

```
[username] ALL=NOPASSWD: /usr/sbin/trafgen
```

Výpis 3.9 obsahuje řádek, který přidáme na konec souboru `/etc/sudoers`, poté při spouštění Trafgen procesu s `root` právy nebudeme nuceni zadávat heslo superuživatele.

Třída CustomProcess.java

Nastavení a spouštění Trafgen procesu probíhá pomocí metod třídy `CustomProcess`. Umístěním kódu odpovídajícím za generování procesu do samostatné třídy zlepšíme přehlednost a čitelnost kódu. Hlavní metody třídy jsou `createProcess()` a `setProcess()`. První z nich odpovídá za spuštění procesu a jeho správné ukončení. Druhá metoda nastavuje spouštěcí příkaz pro nástroj Trafgen. Třída `CustomProcess` je součástí balíčku všech přídavných modulů.

Tvorba konfiguračního souboru Trafgen

Generování konfiguračního souboru Trafgen probíhá po spuštění modulu. Opakovaným spuštěním modulu se obsah konfiguračního souboru přepíše. Třída `Ssdp.java` obsahuje metodu `createPacket()`, která vygeneruje konfigurační soubor na základě nastavených parametrů a uloží do kořenové složky nástroju Trafgen. Jelikož Trafgen neobsahuje implementaci SSDP protokolu pomocí hlavičkové funkce, daný protokol byl vyjádřen v hexadecimálním formátu, viz výpis 3.10.

Výpis 3.10: Konfigurační soubor útoku SSDP DoS attack.

```
{
eth(da=00:50:56:f2:1a:bf, sa=3c:f0:11:d4:2f:dd),
```

```
ipv4(id=136, ttl=64, da=192.168.0.1, sa=192.168.65.134),
udp(sport=drnd(49500,52000), dport=1900, csum=0),
0x4d, 0x2d, 0x53, 0x45, 0x41, 0x52, 0x43, 0x48, 0x20, 0x2a, 0x20, 0x48,
0x54, 0x54, 0x50, 0x2f, 0x31, 0x2e, 0x31, 0x0d, 0x0a, 0x48, 0x4f, 0x53,
0x54, 0x3a, 0x20, 0x32, 0x33, 0x39, 0x2e, 0x32, 0x35, 0x35, 0x2e, 0x32,
0x35, 0x35, 0x2e, 0x32, 0x35, 0x30, 0x3a, 0x31, 0x39, 0x30, 0x30, 0x0d,
0x0a, 0x4d, 0x41, 0x4e, 0x3a, 0x20, 0x22, 0x73, 0x73, 0x64, 0x70, 0x3a,
0x64, 0x69, 0x73, 0x63, 0x6f, 0x76, 0x65, 0x72, 0x22, 0x0d, 0x0a, 0x4d,
0x58, 0x3a, 0x20, 0x31, 0x0d, 0x0a, 0x53, 0x54, 0x3a, 0x20, 0x73, 0x73,
0x64, 0x70, 0x3a, 0x61, 0x6c, 0x6c, 0x0d, 0x0a, 0x55, 0x53, 0x45, 0x52,
0x2d, 0x41, 0x47, 0x45, 0x4e, 0x54, 0x3a, 0x20, 0x47, 0x6f, 0x6f, 0x67,
0x6c, 0x65, 0x20, 0x43, 0x68, 0x72, 0x6f, 0x6d, 0x65, 0x2f, 0x38, 0x30,
0x2e, 0x30, 0x2e, 0x33, 0x39, 0x38, 0x37, 0x2e, 0x31, 0x30, 0x30, 0x20,
0x57, 0x69, 0x6e, 0x64, 0x6f, 0x77, 0x73, 0x0d, 0x0a, 0x0d, 0x0a
}
```

3.2.1 Testování modulu SSDP DoS attack

Testování modulu probíhalo s využitím dvou počítačů, kdy jeden byl použit pro odesílání paketů, druhý pro přijetí paketů z reflexního zařízení. Oba počítače jsou připojené ke stejné síti, primárním účelem druhého počítače je ukázat, že skutečně dochází k obdržení nežádoucích paketů z reflexního zařízení.

Zapojení testovací sítě je znázorněno na obrázku 3.3. Podle zmíněného obrázku `Router2` slouzi jako další UPnP zařízení v síti a je spojený s `Router1` vazbou LAN - LAN.

Hostitelský počítač `PC1` je vybaven čtyřjádrovým procesorem Intel Core i7-8565U o frekvenci 1.8 GHz s 16 GB pamětí RAM. Operační systém je Windows 10 Home x64. Na počítači `PC1` je spuštěný virtuální stroj CentOS 8, ze kterého jsou posílány SSDP žádosti přes aplikaci JMeter.

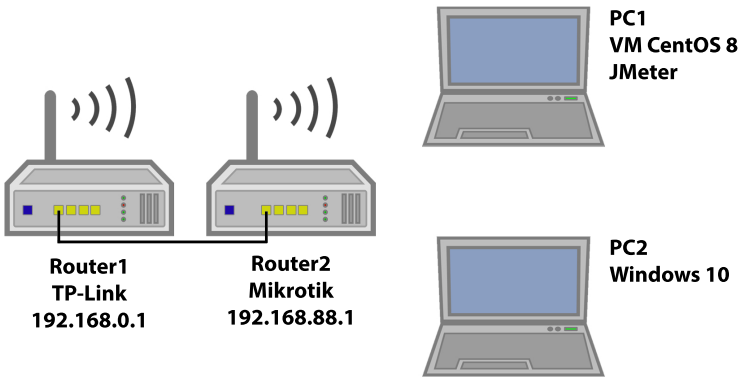
Počítač `PC2` je vybaven čtyřjádrovým procesorem AMD A10-4600M o frekvenci 2.3 GHz s 8GB pamětí RAM. Operační systém je Windows 10 Pro x64. Počítač `PC2` odpovídá za přijetí SSDP odpovědi od reflexních zařízení.

Nastavení SSDP útoku je znázorněno na obrázku 3.4. V tomto případě IP adresy počítačů jsou následující:

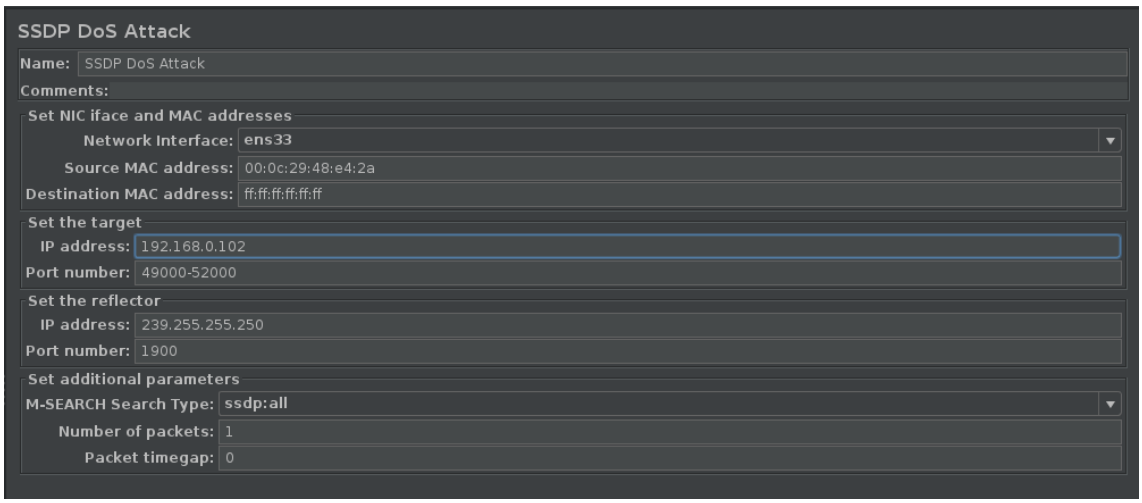
- **PC1**: 192.168.0.101
- **PC2**: 192.168.0.102

Oba počítače jsou připojení k routeru `Router1` pomocí bezdrátového spojení.

Odesláním paketu na všesměrovou MAC adresu docílíme toho, že každé UPnP zařízení v síti pošle odpověď. Na snímku 3.5 je vidět jednu zprávu `M-SEARCH` spolu s odpověďmi od dvou UPnP routerů. Snímek obrazovky byl udělaný na počítači `PC2`.



Obr. 3.3: Schéma zapojení testovací sítě SSDP útoku.

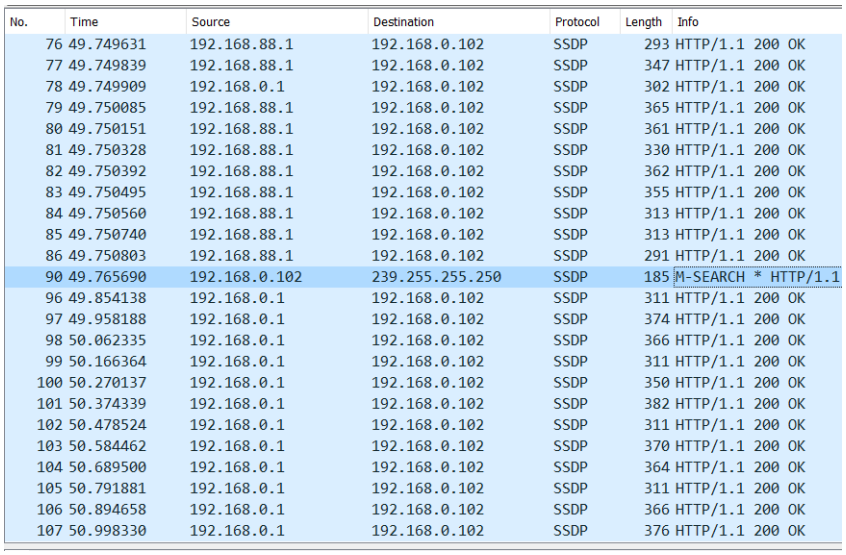


Obr. 3.4: Nastavení útoku SSDP DoS attack.

Pro naslouchání komunikace byl využit program Wireshark. Hodnota amplifikace (BFS) v tomto případě je 42.2.

Testování výkonnosti modulu

Hlavním cílem SSDP útoku je zaplavení oběti nežádoucími pakety, proto je třeba se vyhnout přerušení provozu UPnP zařízení (v tomto případě routeru). Nastavení testovací sítě je stejné jako na obrázku 3.3. V průběhu testování bylo zjištěno, že nastavení prodlevy mezi pakety ovlivňuje datový tok který je směřován na oběť. Je to způsobeno tím, že při zatížení UPnP zařízení `M-SEARCH` pakety, ne všechny požadavky mohou být zpracované. Tabulka 3.2 ukazuje závislost mezi prodlevou a datovým tokem a vyjadřuje hodnoty maximální zátěže na straně oběti.



No.	Time	Source	Destination	Protocol	Length	Info
76	49.749631	192.168.88.1	192.168.0.102	SSDP	293	HTTP/1.1 200 OK
77	49.749839	192.168.88.1	192.168.0.102	SSDP	347	HTTP/1.1 200 OK
78	49.749909	192.168.0.1	192.168.0.102	SSDP	302	HTTP/1.1 200 OK
79	49.750085	192.168.88.1	192.168.0.102	SSDP	365	HTTP/1.1 200 OK
80	49.750151	192.168.88.1	192.168.0.102	SSDP	361	HTTP/1.1 200 OK
81	49.750328	192.168.88.1	192.168.0.102	SSDP	330	HTTP/1.1 200 OK
82	49.750392	192.168.88.1	192.168.0.102	SSDP	362	HTTP/1.1 200 OK
83	49.750495	192.168.88.1	192.168.0.102	SSDP	355	HTTP/1.1 200 OK
84	49.750560	192.168.88.1	192.168.0.102	SSDP	313	HTTP/1.1 200 OK
85	49.750740	192.168.88.1	192.168.0.102	SSDP	313	HTTP/1.1 200 OK
86	49.750803	192.168.88.1	192.168.0.102	SSDP	291	HTTP/1.1 200 OK
90	49.765690	192.168.0.102	239.255.255.250	SSDP	185	M-SEARCH * HTTP/1.1
96	49.854138	192.168.0.1	192.168.0.102	SSDP	311	HTTP/1.1 200 OK
97	49.958188	192.168.0.1	192.168.0.102	SSDP	374	HTTP/1.1 200 OK
98	50.062335	192.168.0.1	192.168.0.102	SSDP	366	HTTP/1.1 200 OK
99	50.166364	192.168.0.1	192.168.0.102	SSDP	311	HTTP/1.1 200 OK
100	50.270137	192.168.0.1	192.168.0.102	SSDP	350	HTTP/1.1 200 OK
101	50.374339	192.168.0.1	192.168.0.102	SSDP	382	HTTP/1.1 200 OK
102	50.478524	192.168.0.1	192.168.0.102	SSDP	311	HTTP/1.1 200 OK
103	50.584462	192.168.0.1	192.168.0.102	SSDP	370	HTTP/1.1 200 OK
104	50.689500	192.168.0.1	192.168.0.102	SSDP	364	HTTP/1.1 200 OK
105	50.791881	192.168.0.1	192.168.0.102	SSDP	311	HTTP/1.1 200 OK
106	50.894658	192.168.0.1	192.168.0.102	SSDP	366	HTTP/1.1 200 OK
107	50.998330	192.168.0.1	192.168.0.102	SSDP	376	HTTP/1.1 200 OK

Obr. 3.5: Testování modulu SSDP DoS attack.

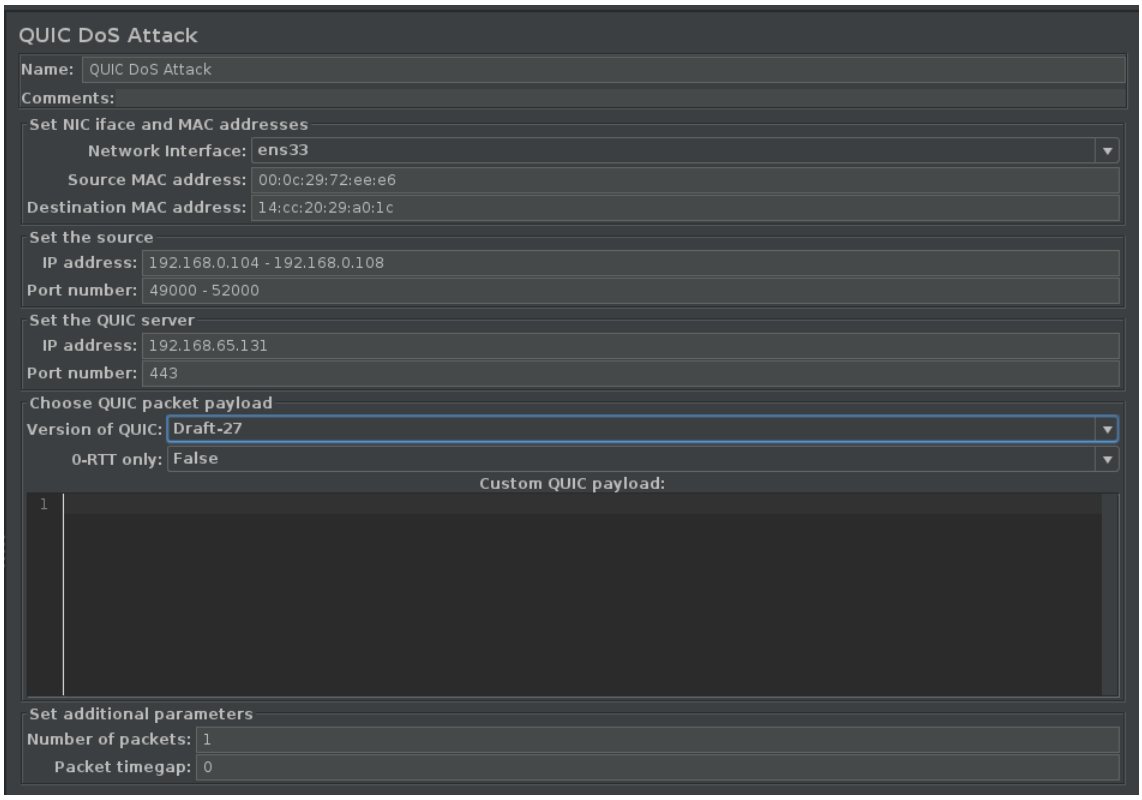
Tab. 3.2: Testování výkonnosti modulu SSDP DoS attack.

Prodleva [ms] / Počet paketů [p]	100 p	1000 p	1000 p	1000 p
bez prodlevy	0,139MB/s	0,258MB/s	0,519MB/s	0,681MB/s
0,5 ms	0,428MB/s	1,51 MB/s	2,35MB/s	2,83MB/s
1 ms	**0,488MB/s**	**2,55MB/s**	**2,79MB/s**	**3,30MB/s**
5 ms	0,461MB/s	1,14MB/s	1,27MB/s	1,22MB/s
10 ms	0,333MB/s	0,658MB/ss	0,688MB/s	0,689MB/s

3.3 Modul QUIC DoS attack

Grafické uživatelské rozhraní modulu je definováno ve třídě QuicBeanInfo.java a je zobrazeno na obrázku 3.6. Grafické prvky jsou rozdělené do jednotlivých skupin tak aby nastavení útoku bylo intuitivní.
Tento modul je rozdělen do pěti skupin (shora dolů):

- Nastavení síťové karty a MAC adres – uživatel definuje síťové rozhraní ze kterého budou posílané pakety a také zdrojovou a cílovou MAC adresu. Většinou platí, že cílová MAC adresa je adresou následujícího zařízení zpracovávající daný paket, např. adresa routeru ke kterému je připojen počítač odesílající pakety.
- Nastavení cíle útoku – definuje se IP adresa a port zařízení na které chceme odpovědi z QUIC serveru směřovat.
- Nastavení QUIC serveru – definuje se IP adresa a port QUIC serveru.
- Nastavení obsahu QUIC paketu – výběr obsahu QUIC paketu ze tří možností.
- Nastavení dalších parametrů – udávají se parametry specifické pro Trafgen.



Obr. 3.6: Grafické uživatelské rozhraní modulu QUIC DoS attack.

Při tvorbě konfiguračního souboru Trafgen je možnost výběru obsahu QUIC paketu. Modul podporuje obsah aktuálně nejpopulárnějších verzí QUIC protokolu: `draft-27`, `Q050`, `Q046`. Součástí balíčku modulu je soubor `QuicPayload.xml` obsahující definici obsahu QUIC paketu. Jelikož QUIC protokol se stále vyvíjí, v okamžiku vzniku nové verze stačí přidat do souboru `QuicPayload.xml` obsah nové verze.

Další možností je výběr 0-RTT paketu, při zvolení této možnosti je ignorována nastavená verze QUIC.

Poslední možností je nastavení vlastního obsahu QUIC paketu, sloužící pro odladování, např. odeslání paketu QUIC s verzi, která není v seznamu modulu QUIC DoS attack. Obsah se definuje ve formátu `Hex Stream`, který je snadno kopírovatelný pomocí programu Wireshark. Při zvolení této možnosti je ignorována nastavená verze QUIC a výběr 0-RTT paketu.

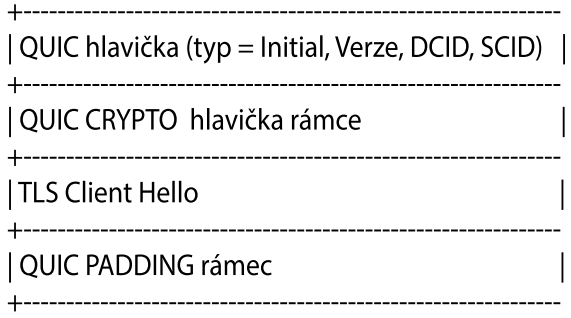
Vygenerovaný konfigurační soubor je vidět po spuštění modulu v záložce `Request` naslouchače `View Results Tree`.

Bezpečnostní analýza QUIC protokolu proti DoS útokům

Před provedením testování QUIC modulu je nejdříve potřeba uvést bezpečnostní techniky které mohou být aplikované proti DoS útokům.

Minimalni délka inicializační zprávy

První inicializační zpráva (obr. 3.7) od klienta musí být nejméně 1200 bajtů. Inicializační zpráva menší než 1200 bajtů je automaticky zahozená serverem. Hlavním důvodem je zabránění amplifikačnímu útoku. V tomto případě útočník využívá přibližně stejně velkou šířku pásma jako server, což dělá tento typ útoku nepraktickým.



Obr. 3.7: QUIC inicializační zpráva klienta.

Prokázání vlastnictví IP adresy

Aby server nevyčerpával svoje zdroje na výpočetní náklady potřebné pro navázání spojení, požádá klienta o prokázání vlastnictví IP adresy. Na požadavek klienta na spojení, server zašle buď `Retry` paket nebo `Initial` paket (inicializační zpráva). V případě `Initial` paketu server zvolí vlastní SCID (Source Connection ID). Všechny následující pakety od klienta musí obsahovat DCID (Destination Connection ID) rovným SCID serveru. V případě `Retry` paketu server navíc vygeneruje a zašle token. Klient musí zduplikovat svůj požadavek a uvést token obdržený od serveru. Nedostatek dané metody je ve přidání dodatečného kola výměny informace (angl. round trip).

Avšak na některých serverech prokázání vlastnictví IP adresy není. Výměna kryptografických klíčů je implicitním důkazem, že klient vlastní IP adresu a spojení skutečně vyžadoval. V tomto případě na první inicializační zprávu od klienta, server musí omezit počet dat který zašle ihned klientovi [8].

3.3.1 Testování modulu QUIC DoS attack

Pro testování modulu QUIC DoS attack byl zvolen server OpenLiteSpeed ve verzi 1.6.10, který obsahuje podporu QUIC protokolu.

Instalace a konfigurace QUIC serveru

Instalace serveru OpenLiteSpeed probíhala na virtuálním počítači s operačním systémem Ubuntu 18.04 Server. Po stažení binární verze serveru stačí rozbalit `.tgz` soubor, přejít do kořenové složky a spustit skript `install.sh`. Všechny závislosti, potřebné pro správnou funkčnost serveru, by se měli nainstalovat automaticky. Server spustíme příkazem viz výpis 3.11.

Výpis 3.11: Příkaz na spouštění OpenLiteSpeed serveru.

```
/usr/local/lsws/bin/lswsctrl start
```

Konfigurace serveru probíhá na adrese `https:\\[IP adresa serveru]:7080` pomocí webového rozhraní. Ve záložce `Listeners` přidáme HTTPS naslouchač na portu 443. Pro zabezpečený přístup na server s podporou QUIC musíme vygenerovat certifikáty. Postup generování certifikátů je následující:

- Vytvořit vlastní certifikační autoritu (CA)
  - Vygenerovat `root` klíč.
  - Pomocí `root` klíče vygenerujeme certifikát podepsaný sám sebou tzv. „self-signed certificate“. Tento certifikát bude vystupovat jako certifikát certifikační autority. Musí být přidán na počítač do sekcí důvěryhodných certifikačních autorit.
- Vytvořit certifikát pro server
  - Vygenerovat soukromý klíč serveru.
  - Vytvořit žádost o podepsání certifikátu tzv. „csr“. Položku `Common Name` vyplnit IP adresou serveru.
  - Vygenerovat certifikát serveru podepsáním `csr` pomocí `root` klíče certifikační autority.

Soukromý klíč serveru a certifikát přidáme do záložky SSL naslouchače HTTPS, jak je zobrazeno na obrázku 3.8.

Posledním krokem je povolit ve firewallu komunikaci na UDP portu 443 (viz výpis 3.12).

Výpis 3.12: Povolení zabezpečeného spojení na UDP portu.

```
sudo ufw allow 443/udp
```

Po úspěšné instalaci a nastavení serveru se v levém horním rohu webové stránky musí zazelenat zámeček jak je to na obrázku 3.9.

Amplifikační útok na instalovaný server byl neúspěšný. Podle obrázku 3.10 server implementuje bezpečnostní techniku prokázání vlastnictví IP adresy. Na inicializační zprávu klienta server odpověděl svojí inicializační zprávou ve které uvedl

vlastní SCID. Všechny následující pakety od klienta musí odkazovat na SCID vybraný serverem, jinak pakety budou zahozené. Klient opakuje inicializační zprávu s novým DCID (Destination Connection ID) rovným SCID serveru.

Testování modulu na různých implementacích QUIC serveru

Jelikož se nepodařilo ukázat efekt amplifikace na serveru OpenLiteSpeed, pro testování byly zvolené další veřejně dostupné implementace QUIC serveru.
IETF QUIC Working Group sdílí ve svém veřejně dostupném GitHub repositáři různé implementace QUIC klientů a serverů. Komunita vývojářů má možnost sdílet svoje implementace a také vyzkoušet implementaci ostatních [13].

QUIC implementace č.1

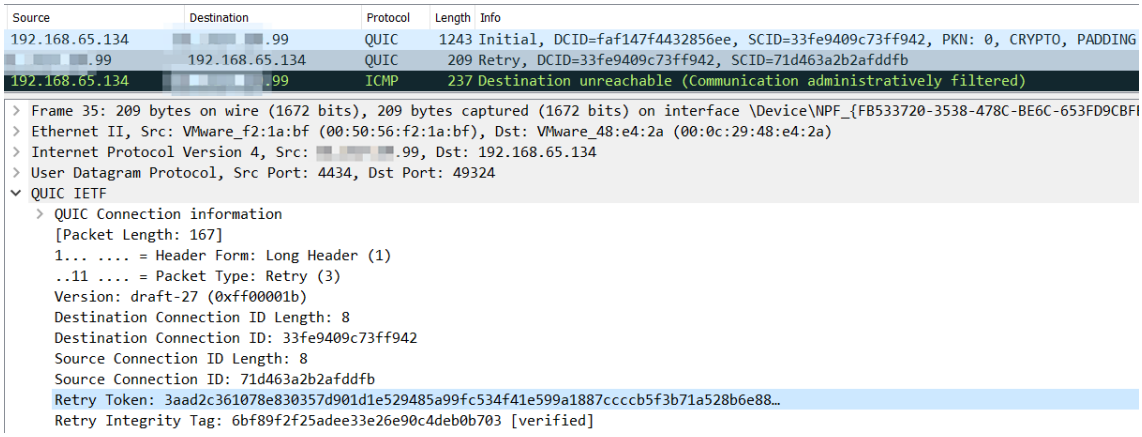
Byla vyzkoušena implementace QUIC protokolu „quicly“ pro H20 server. Na každý dotaz klienta vznikala amplifikace 3x větší než velikost inicializační zprávy klienta. Podle obrázku 3.11 server odpovídá dvěma Handshake pakety, v realitě první paket je typu `Initial` a druhý `Handshake`. Wireshark ne vždy může správně rozeznat QUIC pakety, což je způsobeno tím, že různé verze Wireshark jsou provázané s různými verzemi QUIC protokolu.

Source	Destination	Protocol	Length	Info
192.168.65.134	[redacted].244	QUIC	1243	Initial, DCID=faf147f4432856ee, SCID=33fe9409c73ff942,
[redacted].244	192.168.65.134	QUIC	1322	Handshake, DCID=33fe9409c73ff942, SCID=0ea771d547628f64
192.168.65.134	[redacted].244	ICMP	590	Handshake, DCID=33fe9409c73ff942, SCID=0ea771d547628f64
[redacted].244	192.168.65.134	QUIC	1322	Handshake, DCID=33fe9409c73ff942, SCID=0ea771d547628f64
[redacted].244	192.168.65.134	QUIC	878	Protected Payload (KP0), DCID=33fe9409c73ff942
192.168.65.134	[redacted].244	ICMP	590	Destination unreachable (Communication administratively
192.168.65.134	[redacted].244	ICMP	590	Destination unreachable (Communication administratively
[redacted].244	192.168.65.134	QUIC	1317	Handshake, DCID=33fe9409c73ff942, SCID=0ea771d547628f64
192.168.65.134	[redacted].244	ICMP	590	Handshake, DCID=33fe9409c73ff942, SCID=0ea771d547628f64

Obr. 3.11: Amplifikace na serveru H20 s podporou QUIC.

QUIC implementace č.2

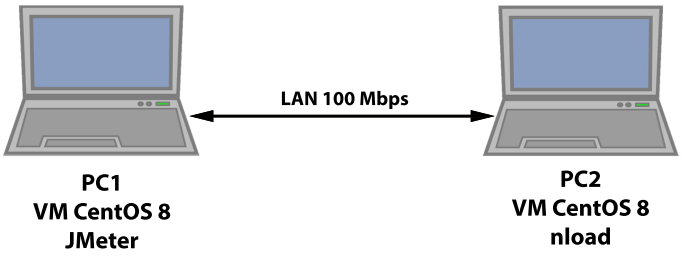
Byla vyzkoušena implementace QUIC protokolu „aioquic“. Na požadavek klienta na spojení server zašle `Retry` paket ve kterém uvede token (obr. 3.13). Klient musí zduplikovat inicializační zprávu ve které uvede token. Amplifikační útok se v tomto případě nezdařil.



Obr. 3.12: Prokázání vlastnictví IP adresy pomoci Retry paketu.

Testování výkonnosti modulu

Testování probíhalo pomocí dvou počítačů spojených Ethernet kabelem. Zapojení testovací sítě je znázorněno na obrázku 3.13.

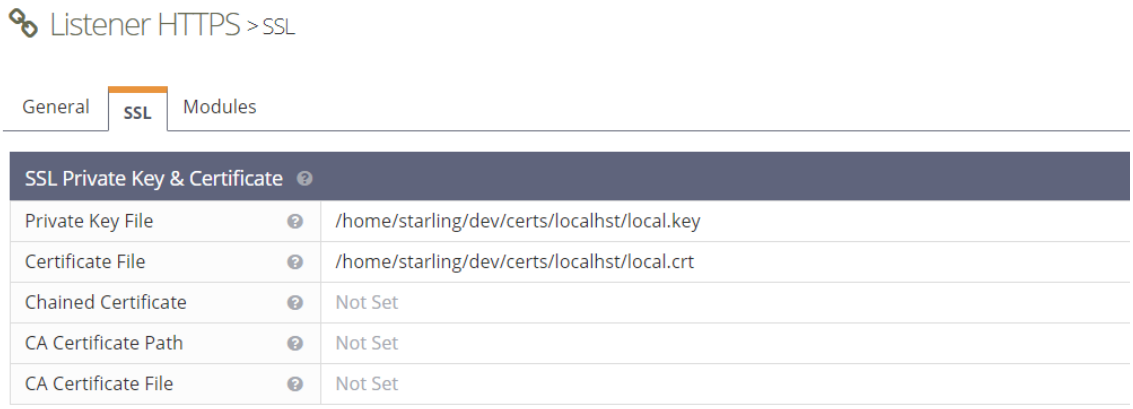


Obr. 3.13: Schéma zapojení testovací sítě QUIC útoku.

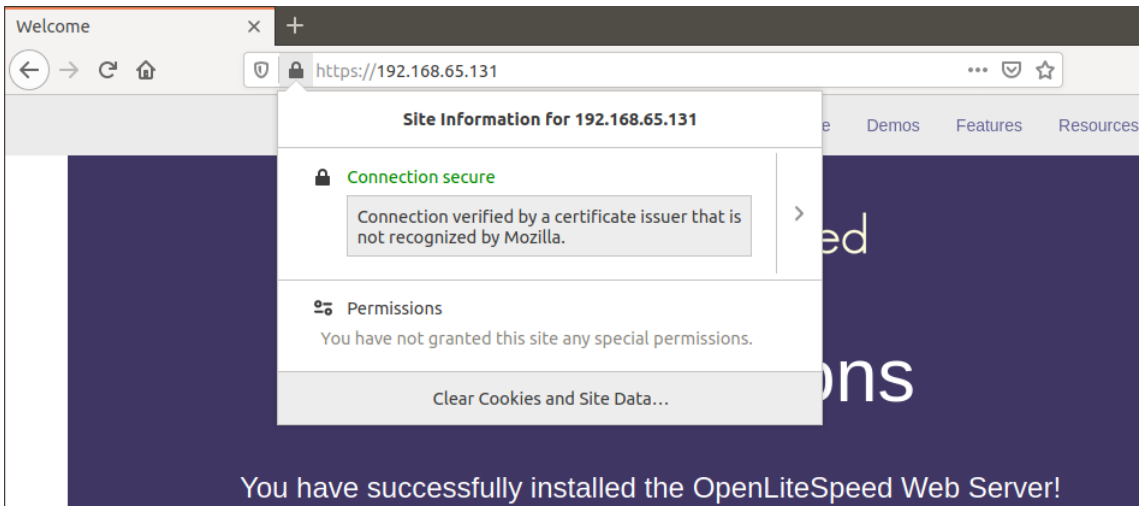
Tabulka 3.3 ukazuje závislost mezi nastaveným obsahem QUIC paketu (angl. payload) a vytížením linky. Pokaždé se vysílalo 1 000 000 paketů s největší možnou rychlostí odeslání. Nastavený počet procesorů virtuálního počítače je 4 CPU.

Tab. 3.3: Testování výkonnosti modulu QUIC DoS attack.

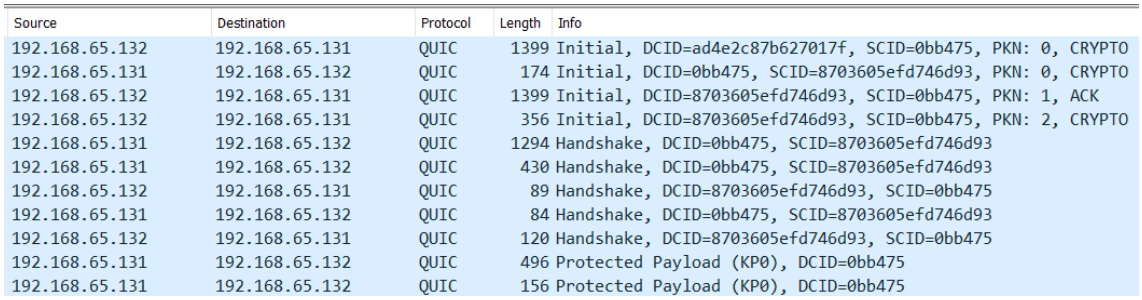
QUIC payload	**Velikost paketu [B]**	**Vytížení linky [Mbit/s]**
Vlastní	60	8,9
0-RTT	300	42,5
Q046	1412	97,3



Obr. 3.8: Nastavení zabezpečeného spojení na serveru OpenLiteSpeed.



Obr. 3.9: Ukázka zabezpečeného spojení na serveru OpenLiteSpeed.



Source	Destination	Protocol	Length	Info
192.168.65.132	192.168.65.131	QUIC	1399	Initial, DCID=ad4e2c87b627017f, SCID=0bb475, PKN: 0, CRYPTO
192.168.65.131	192.168.65.132	QUIC	174	Initial, DCID=0bb475, SCID=8703605efd746d93, PKN: 0, CRYPTO
192.168.65.132	192.168.65.131	QUIC	1399	Initial, DCID=8703605efd746d93, SCID=0bb475, PKN: 1, ACK
192.168.65.132	192.168.65.131	QUIC	356	Initial, DCID=8703605efd746d93, SCID=0bb475, PKN: 2, CRYPTO
192.168.65.131	192.168.65.132	QUIC	1294	Handshake, DCID=0bb475, SCID=8703605efd746d93
192.168.65.131	192.168.65.132	QUIC	430	Handshake, DCID=0bb475, SCID=8703605efd746d93
192.168.65.132	192.168.65.131	QUIC	89	Handshake, DCID=8703605efd746d93, SCID=0bb475
192.168.65.131	192.168.65.132	QUIC	84	Handshake, DCID=0bb475, SCID=8703605efd746d93
192.168.65.132	192.168.65.131	QUIC	120	Handshake, DCID=8703605efd746d93, SCID=0bb475
192.168.65.131	192.168.65.132	QUIC	496	Protected Payload (KP0), DCID=0bb475
192.168.65.131	192.168.65.132	QUIC	156	Protected Payload (KP0), DCID=0bb475

Obr. 3.10: Komunikace serveru OpenLiteSpeed s podporou QUIC protokolu.

Závěr

Cílem bakalářské práce bylo ověření funkčnosti zátěžového testeru a jeho rozšíření o přídavné moduly DoS útoků zaměřených na amplifikační útoky. Teoretická část práce pojednává o problematice DoS útoků, jejich základní klasifikaci. Některé popsané DoS útoky jsou součástí zátěžového testeru.

Druhá kapitola se věnuje problematice zátěžového testování a také aplikaci JMeter, na bázi které je postavený testovaný systém. Kapitola obsahuje popis základních elementů aplikace a jejich hierarchii při spouštění testovacího plánu. Vedle toho je zmíněn nástroj Trafgen, který slouží jako generátor síťového provozu.

Dalším bodem bylo ověření funkčnosti zátěžového testeru. Nejprve bylo provedeno testování s využitím mezilehlého síťového prvku, kde byla zjištěna jeho maximální možná zátěž. Dále následovalo ověření funkčnosti systému jako celku a to propojením externích zásuvných modulů spolu s nativními elementy aplikace JMeter. Výsledky testovacích případů jsou doprovázené tabulkami a grafy, které znázorňují funkčnost systému. V průběhu testování bylo zjištěno několik chyb v implementaci modulu zátěžového testeru, které byly okomentované v práci.

Třetí kapitola se věnuje samotnému vytvoření modulu DoS útoku. Na začátku kapitoly je popis konfigurace vývojového prostředí, což zahrnuje podrobný návod k instalaci aplikace JMeter a nástroje Trafgen. Vedle toho je zmíněn způsob pro sestavení libovolného konfiguračního souboru Trafgen. Realizované amplifikační moduly DoS útoku jsou „SSDP DoS attack“ a „QUIC DoS attack“. Popis modulu SSDP DoS attack obsahuje popis základních tříd které jsou podobné i pro další modul. Testováním modulu SSDP DoS attack bylo zjištěno, že zařízení s podporou UPnP protokolu jsou náchylné vůči amplifikačním DoS útokům. Provedený výkonnostní test ukazuje závislost generované zátěže od reflexních zařízení vůči prodlevě mezi žádostmi od klienta.
Následně kapitola obsahuje popis modulu QUIC DoS attack. Uvedené jsou výsledky testování QUIC protokolu z hlediska bezpečnosti proti DoS útokům. Podle výsledků provedených testů se ukázalo, že komunikační protokol QUIC je dostatečně odolný vůči DoS útokům. QUIC protokol stále vyvíjí a není plně standardizovaný, vyskytují se proto varianty a implementace QUIC protokolu které jsou vůči amplifikačnímu typu útoku náchylné.

Literatura

[1] BHATIA S., BEHAL S. a AHMED I. *Distributed Denial of Service Attacks and Defense Mechanisms: Current Landscape and Future Directions*. In: Conti M., Somani G., Poovendran R. (eds) Versatile Cybersecurity. Advances in Information Security, vol 72. Springer, Cham, 2018. ISBN 978-3-319-97643-3.

[2] HOFFMAN, CHRIS. *What Is a Botnet?* How-To Geek [online]. 2016 [cit. 03. 11. 2019]. Dostupné z URL: <`https://www.howtogeek.com/183812/htg-explains-what-is-a-botnet/`>.

[3] CISA. *UDP-Based Amplification Attacks* [online]. 2014, poslední aktualizace 02. 03. 2018 [cit. 2019. 11. 03]. Dostupné z URL: <`https://www.us-cert.gov/ncas/alerts/TA14-017A`>.

[4] A10 Networks. *Understanding DDoS Attacks* [online]. 2018, poslední aktualizace 25. 09. 2018 [cit. 2019. 11. 03]. Dostupné z URL: <`https://www.a10networks.com/blog/understanding-ddos-attacks/`>.

[5] Radware. *DDoS Handbook* [online]. [cit. 15. 11. 2019]. Dostupné z URL: <`https://www.radware.com/ddoshandbook/`>.

[6] The Cloudflare Blog. *Stupidly Simple DDoS Protocol (SSDP) generates 100 Gbps DDoS* [online]. 2017, poslední aktualizace 28. 06. 2017 [cit. 20. 01. 2019]. Dostupné z URL: <`https://blog.cloudflare.com/ssdp-100gbps/`>.

[7] GHEDINI, A. *The Road to QUIC*. [online]. 2018, poslední aktualizace 26. 07. 2017 [cit. 10. 05. 2020]. Dostupné z URL: <`https://blog.cloudflare.com/the-road-to-quic/`>.

[8] IYENGAR, J. a THOMSON M. *QUIC: A UDP-Based Multiplexed and Secure Transport*. [online]. 2020, poslední aktualizace 21. 02. 2020 [cit. 05. 06. 2020]. Dostupné z URL: <`https://tools.ietf.org/html/draft-ietf-quic-transport-27`>.

[9] FISCHLIN, M. a GÜNTHER F. *Replay Attacks on Zero Round-Trip Time: The Case of the TLS 1.3 Handshake Candidates*. [online]. 2020, poslední aktualizace 03. 07. 2017 [cit. 05. 06. 2020]. Dostupné z URL: <`https://ieeexplore.ieee.org/document/7961952`>.

[10] Load Storm. *Load Testing Metrics Explained* [online]. [cit. 23. 11. 2019]. Dostupné z URL: <`https://loadstorm.com/load-testing-metrics/`>.

[11] Apache Jmeter. *Apache JMeter™* [online]. [cit. 25. 11. 2019]. Dostupné z URL: <https://jmeter.apache.org/>.

[12] BORKMANN, D. *Trafgen – a fast, multithreaded network packet generator*. [online] [cit. 30. 11. 2019]. Dostupné z URL: <http://man7.org/linux/man-pages/man8/trafgen.8.html>.

[13] IETF QUIC Working Group. *QUIC implementations*. [online] [cit. 05. 05. 2020]. Dostupné z URL: <https://github.com/quicwg/base-drafts/wiki/Implementations>.

Seznam symbolů, veličin a zkratek

BAF	Bandwidth Amplification Factor
C&C	Command and Control
CPU	Central Processing Unit
DCID	Destination Connection ID
DDoS	Distributed Denial of Service
DHT	Distributed Hash Table
DNS	Domain Name System
DoS	Denial of Service
GUI	Graphical User Interface
FTP	File Transfer Protocol
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
LAN	Local Area Network
PPS	packets per second
QUIC	Quick UDP Internet Connections
RAM	Random Access Memory
SCID	Source Connection ID
SOAP	Simple Object Access Protocol
SSDP	Simple Service Discovery Protocol
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UPnP	Universal Plug and Play
XML	eXtensible Markup Language
0-RTT	zero round-trip time

Seznam příloh

A Obsah přiloženého CD

Přiloženy médium obsahuje elektronickou verzi bakalářské práce spolu se zdrojovými kódy přídavných modulů do aplikace JMeter. Takže médium obsahuje `.jar` soubor s kompilovanými moduly, který stačí naimportovat do složky externích modulů aplikace JMeter.

- `/` ..... kořenový adresář přiloženého CD
  - `modules_src.zip` ..... soubor se zdrojovými kódy modulů
    - `org`
      - `apache`
        - `jmeter`
          - `ssdp`
            - `Ssdp.java`
            - `SsdpBeanInfo.java`
            - `SsdpResources.properties`
            - `CustomProcess.java`
          - `quic`
            - `QuicSampler.java`
            - `QuicSamplerBeanInfo.java`
            - `QuicSamplerResources.properties`
            - `QuicPayload.xml`
            - `CustomProcess.java`
  - `ApacheJMeter_custom.jar` ..... soubor s DoS amplifikačními moduly JMeter
  - `Zatezovy-tester-xshpak01.pdf` ..... elektronická verze práce