VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2019 Jan Bachorec

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

VIZUALIZACE FILTRACE DDOS ÚTOKŮ
VISUALIZATION OF DDOS ATTACKS MITIGATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

Jan Bachorec

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Zdeněk Martinásek, Ph.D.

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Informační bezpečnost**
Ústav telekomunikací

Student: Jan Bachorec ***ID:*** 195148
Ročník: 3 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Vizualizace filtrace DDoS útoků

POKYNY PRO VYPRACOVÁNÍ:

Cílem bakalářské práce je návrh a implementace vhodného grafického uživatelského rozhraní pro systém filtrace kybernetických útoků cílených na odepření služeb (Denial of service (DoS). Grafické uživatelské rozhraní bude mít formu interaktivní webové stránky a umožní vhodnou analýzu průběhu DDoS útoku (čas, síla, typ), dopadu na legitimní provoz (celý systém, webové servery) a činnosti filtračních serverů. V teoretické části analyzujte vhodné šablony pro tvorbu rozhraní včetně definice sledovaných parametrů. Seznamte se s experimentálním pracovištěm. V rámci bakalářské práce navrhněte a implementujte grafické uživatelské rozhraní analyzující záplavový útok SYN Flood, UDP Flood, ICMP Flood, DNS Flood a HTTP Flood na experimentálním pracovišti.

DOPORUČENÁ LITERATURA:

[1] GASSTON, Peter. The modern Web: multi-device Web development with HTML5, CSS3, and JavaScript. San Francisco: No Starch Press, 2013. ISBN 15-932-7487-4.

[2] BHUYAN, Monowar H.; BHATTACHARYYA, D. K.; KALITA, Jugal K. An empirical evaluation of information metrics for low-rate and high-rate DDoS attack detection. Pattern Recognition Letters, 2015.

Termín zadání: 1.2.2019 ***Termín odevzdání:*** 27.5.2019

Vedoucí práce: Ing. Zdeněk Martinásek, Ph.D.
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Bakalářská práce se zabývá tvorbou a implementací grafického uživatelského rozhraní pro systém filtrace kybernetických útoků cílených na odepření přístupu ke službě. Toto rozhraní je ve formě interaktivní webové stránky. Umožňuje přehledně zobrazit základní parametry DDoS útoků, jakými jsou délka trvání, objem síťového provozu. Dále umí graficky interpretovat dopady DDoS útoků na jednotlivá zařízení pracoviště a vizualizovat využití jejich výpočetních kapacit. Například zatížení procesoru a paměti RAM. Grafické uživatelské rozhraní je implementováno na zařízení experimentálního pracoviště VUT. V teoretické části jsou rozebrány DDoS útoky a možnosti obrany vůči nim. Součástí je analýza technologií, jejichž použití bylo zvažováno v rámci praktické části bakalářské práce. Součástí práce je i popis experimentálního pracoviště VUT.

KLÍČOVÁ SLOVA

DDoS, vizualizace, Node.js, HTML, JavaScript, Chart.js, graf

ABSTRACT

Bachelor's thesis deals with the development and implementation of graphical user interface for the system of filtration denial of service cyber attacks. Graphical user interface is in the form of interactive website. It allows users to clearly display basic DDoS attack parameters such as duration, network traffic. Furthermore, it can graphically interpret the impacts of DDoS attacks on individual workplace devices and visualize the use of their computing capabilities. For example, CPU and RAM load. The graphical user interface is implemented on the devices of BUT experimental workplace. In the theoretical part, DDoS attacks and possibilities of defense against them are discussed. The part of the thesis is dedicated to an analysis of technologies whose use was considered within the practical part of the thesis. The part of the thesis also includes a description of the BUT experimental workplace.

KEYWORDS

DDoS, visualisation, Node.js, HTML, JavaScript, Chart.js, graph

BACHOREC, Jan. *Vizualizace filtrace DDoS útoků*. Brno, 2019, 53 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Zdeněk Martinásek, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Vizualizace filtrace DDoS útoků“ 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 Ing. Zdeňkovi Martináskovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této bakalářské práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno ....

podpis autora

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

Obsah

Seznam obrázků

Úvod

V posledních letech došlo k obrovskému pokroku na poli výpočetní techniky. Komunikační sítě se používají ve všech odvětvích lidské činnosti. Velká část společnosti denně používá informační systémy propojené pomocí Internetu. Neděje se tomu tak už jen prostřednictvím počítačů. Díky trendu IoT (Internet of Things) se stanicí v Internetu stávají všemožné přístroje. Uzlem celosvětové sítě je tak například chytrý ovladač ústředního topení nebo televize.

Velké nasazení propojených systémů často dokáže zjednodušit život, například vzdálená komunikace, platby, sdílení informací. Avšak jsou zde i stinné stránky. Různé skupiny a organizace se snaží prosazovat své zájmy, ať už ekonomické či politické, narušováním funkčnosti těchto informačních systémů, které nás v současnosti obklopují. Někdy tak činí pomocí sofistikovaných kybernetických útoků. Velmi často však sáhnou k primitivnímu, leč velmi účinnému řešení. Tím je DoS (Odepření služby z anglického Denial of Service) útok. K jeho provedení v nejjednodušší formě stačí pouze zahltit síť nežádoucím provozem.

Jen v České republice došlo například v minulém roce k DDoS (Distributed Denial of Service) útoku na volební weby [22]. Názorně vidíme, že taková hrozba najednou získává i politické rozměry.

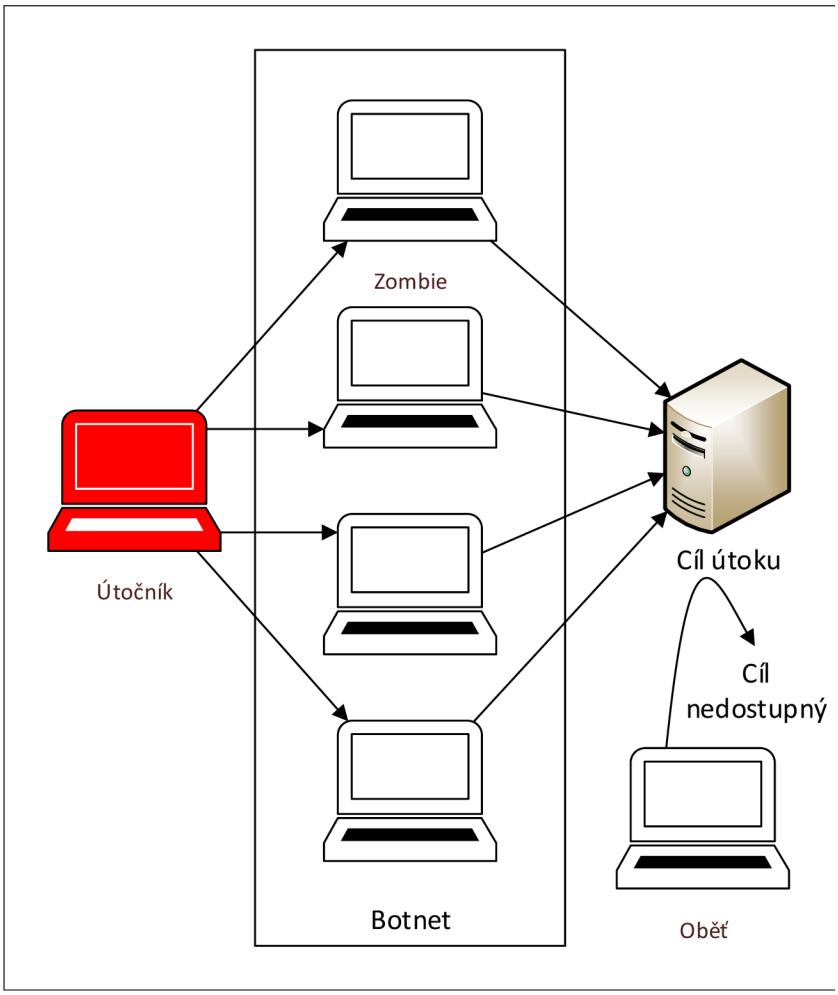
Tato práce se zabývá vizualizací vybraných dat z prostředků technické obrany vůči jedné konkrétní skupině útoků, jejímž cílem je odepření přístupu ke službě. Souhrnně se zmíněný typ hrozby označuje anglickou zkratkou DoS.

Hlavním cílem práce je návrh a realizace projektu grafického rozhraní, pomocí něhož by uživatel mohl rychle a efektivně analyzovat dopady útoku na síťovou infrastrukturu. Vedlejším cílem je pak seznámení se školním experimentálním pracovištěm pro simulaci těchto útoků. V následujících kapitolách práce budou představeny typy útoků na odepření přístupu ke službě. Způsoby, jakými lze těmto rizikům předcházet, zejména pak jejich filtrováním. Rozbor současných možností sběru dat z filtračních zařízení. Část textu je věnována výběru konkrétních parametrů pro následné zpracování, dále pak popisu tvorby prostředí pro vizuální interpretaci těchto dat za pomocí webových technologií HTML (Hypertext Markup Language), CSS (Cascading Style Sheets) a JavaScript. Zhodnoceny jsou možnosti využití šablon webových stránek pro tvorbu finální vizualizace.

1 Teoretická část bakalářské práce

1.1 Útoky cílené na odepření přístupu ke službě

Cílem této hrozby je znemožnit uživateli přístup k nějaké službě či síti. Obecně se tak děje zahlcením cílového systému takovým provozem, který není tento systém schopen obsloužit[1], a proto dojde k jeho přetížení. To způsobí odepření přístupu ke službě uživateli. Obětí takového útoku se může stát kdokoliv. Často to jsou webové stránky různých institucí s vysokou návštěvností. To proto, že je útok zaměřen na působení ztráty cíli jeho dočasnou nedostupností, nikoliv na ukradení dat nebo získání neoprávněného přístupu. Motivací pro takový útok může být konkurenční boj, vydírání, kdy útočník požaduje nějakou odměnu za ukončení útoku a tím umožnění přístupu legitimním uživatelům.



Obr. 1.1: Princip DDoS útoku při zneužití botnetu.

Pro uskutečnění některých typů DoS útoku je nutné generovat velké množství dotazů na nějaký cílový server. Proto se rozlišuje distribuovaný útok na odepření přístupu ke službě Distributed Denial of Service (DDoS) 1.1. Útočník zde pro generování škodlivého provozu nevyužívá pouze jednoho vlastního počítače, nýbrž celé sítě na sobě často nezávislých systémů [1]. Tyto sítě se nazývají „botnety“[1]. Hovoří se zde zpravidla o zařízeních infikovaných nějakým škodlivým softwarem, zneužívaných k útokům bez vědomí jejich oprávněných uživatelů. Může se jednat o počítače, telefony, ale i jakékoliv jiné přístroje s připojením k Internetu.

1.1.1 Typy útoků na odepření přístupu ke službě

Útoky DoS a DDoS jsou jednotlivými zdroji děleny do podskupin. Práce [2] rozlišuje dva základní útoky DoS. Prvním z nich je útok hrubou silou. Oběť je zahlcena intenzivním provozem. Následně dojde ke zpomalení, či úplnému zastavení běžné činnosti cílového systému. Druhou možností pro útočníka je zneužití známé zranitelnosti, například generováním specifických požadavků na server. Tímto jednáním je pak dosaženo, podobně jako v prvním případě, narušení činnosti cílového systému.

Podrobnější dělení nalezneme například na webové stránce [3]. Ta rozlišuje čtyři základní typy. První z možností je rozdělení dle vrstev referenčního síťového modelu ISO/OSI, na kterých probíhají, zejména zde budou zmíněny útoky na aplikační vrstvě. Další možností je dělit hrozby na základě velkého objemu generovaného síťového provozu. Pro třetí skupinu je charakteristické zaměření na určité vlastnosti některých síťových protokolů. Poslední skupina je poměrně specifická, zneužívá nějaké chyby typu Zero-day[2].

Útoky na aplikační vrstvě

Velmi často se zde útočí za pomocí protokolů HTTP/HTTPS (Hypertext Transfer Protocol, Hypertext Transfer Protocol Secured), určených ke komunikaci s webovými servery. Jedním z důvodů je fakt, že různé obranné mechanismy, například firewally, právě tento typ komunikace většinou povolí. Jako příklad je zde uveden útok Slowloris.

Jedná se o přesně cílený útok čistě na webový server oběti. Během něj nedochází k narušení činnosti jiných služeb, ani komunikace na ostatních portech. Útočník na server posílá nekompletní HTTP žádosti, kterými otevírá paralelně nová spojení. Současně ta již běžící udržuje aktivní. Když je vygenerován dostatek souběžných

[1] Síť zařízení, připojených k Internetu, které jednají autonomně nebo automaticky.

[2] Zero-day je nově objevená, neznámá zranitelnost, z její podstaty se nelze bránit pomocí bezpečnostních záplat

spojení, vyčerpá se počet povolených aktivních spojení na serveru a ten pak následující požadavky začne odmítat. Dále tyto hrozby využívají známých chyb serverů, jako jsou Apache, MS ISS a další.

Útoky zneužívající vlastností některých protokolů

Obvykle zneužívanými vlastnostmi jsou pro tuto skupinu například způsoby, jakými se ve spojovaných protokolech navazuje spojení. Nejznámější je útok typu SYN flood. Jelikož je jedním ze dvou útoků, na jejichž zkoumání je tato práce přímo zaměřená, bude jeho popisu věnována samostatná podkapitola.

Charakteristickým zástupcem je i zranitelnost známá jako „Ping of Death“. Pro její realizaci se používá protokol ICMP (Internet Control Message Protocol). Útočník zašle oběti ping o velikosti větší než 65535 bajtů. Ta není schopna takový požadavek zpracovat a dojde k pádu systému.

Patří sem i takzvaný „Smurf attack“, při něm útočník rozesílá ICMP pakety s podvrženou zdrojovou adresou. Tou je adresa oběti, které pak ostatní zařízení v síti odpovídají, čímž ji zahltí.

Záplavové útoky

V tomto případě je síť oběti zahlcena obrovským množstvím paketů, čímž je dosaženo využití celé šířky pásma této sítě a tím i odepření přístupu ke službě poskytované obětí. Do této kategorie patří útok typu UDP (User Datagram Protocol) flood, který bude podrobně rozebrán ve vlastní podkapitole.

Dále sem patří ICMP flood. Při něm je útočníkem generováno množství Echo Request paketů, aniž by útočník čekal na odpověď. Tím je zahlcena jak celá šířka pásma sítě oběti, tak i výpočetní kapacity oběti samotné.

Zero-day útoky

Obecně tyto hrozby definuje nemožnost jejich opakování. V oblasti DoS jsou zero-day dle zdroje [23] chápány jako útoky zneužívající konkrétní techniky a protokoly, za účelem působení odepření přístupu legitimnímu uživateli, v praxi poprvé.

Měřitelné parametry DDoS útoků

Každý DoS a DDoS útok ke svému fungování potřebuje nějaká zařízení, schopná vzájemně komunikovat prostřednictvím sítí. V jeho průběhu je generován síťový provoz. Problém detekce útoku na odepření přístupu ke službě je problémem odlišení parametrů provozu běžného od provozu generovaného samotným útokem.

V rámci počítačových sítí jsou data posílána ve formě paketů. Kdy každý paket má určitou velikost. Síťovému provozu tak jde přiřadit rychlost přenosu dat, charakterizovanou jako poměr množství dat přenesených za jednotku času. Během různých útoků se bude dle typu útoku lišit mohutnost generovaného provozu v závislosti na druhu probíhajícího útoku. Pokud například nebude kapacita síťového provozu generovaného útokem konstantní, bude její průběh vhodné znázorňovat graficky.

Další veličinou charakterizující provoz na síti je šířka pásma. V počítačové síti je definovaná jako maximální rychlost provozu v rámci sítě. Podle typu útoku se bude měnit využití šířky pásma a v rámci některých bude docházet i k úplnému vyčerpání její kapacity.

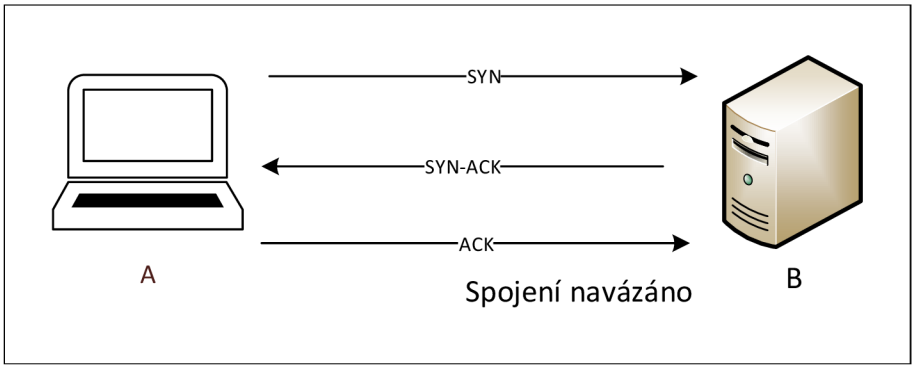
V běžném provozu dochází za jistých okolností, například pokud provoz převýší šířku pásma nějakého zařízení v síti, k zahazování paketů. Jednou z možností, jak v rámci DoS útoku odepřít přístup legitimním uživatelům, je způsobit daným útokem zahazování paketů těchto legitimních uživatelů. Proto i množství zahazovaných paketů je parametrem, jehož sledování je vhodné zvážit v rámci dané problematiky.

Jelikož stanice zapojené v síti potřebují ke zpracování dat přijatých ze sítě, či k zasílání dat na jiné stanice prostřednictvím sítě určité prostředky výpočetní kapacity, bude během útoků možné sledovat určité změny ve vytížení těchto stanic. Zejména v oblasti využití kapacity procesoru a operační paměti. V rámci praktické části této práce bude proto vhodné vytvořeným technickým řešením sbírat a vyhodnocovat i tyto hodnoty.

1.1.2 Vybrané útoky pro praktickou část bakalářské práce

Text se zabývá pěti konkrétními útoky, jejichž metrika budou graficky znázorněna a interpretována v rámci praktické části práce.

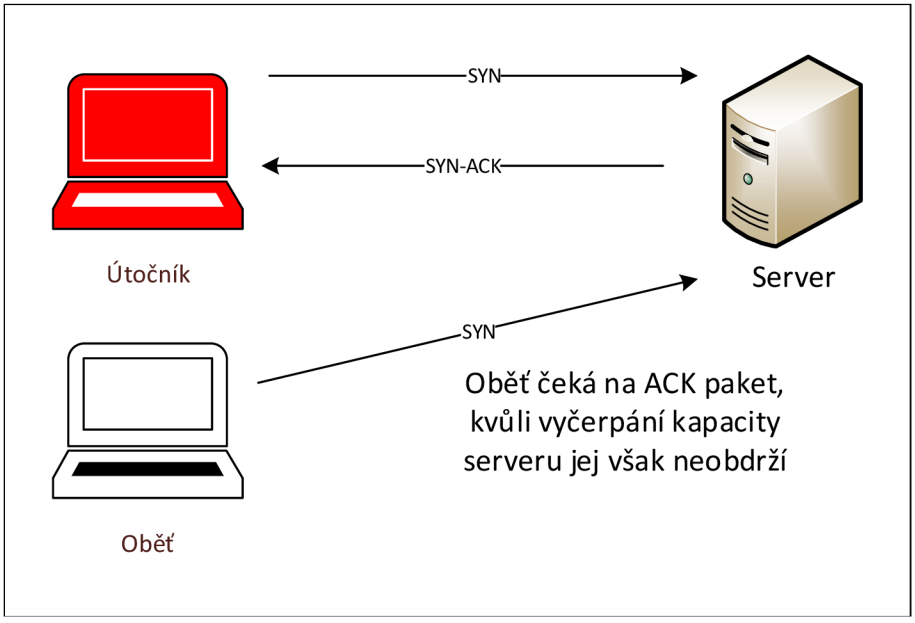
Útok typu SYN Flood



Obr. 1.2: Navazování spojení v protokolu TCP.

Pro uskutečnění SYN Flood útoků je využívána implementace protokolu TCP (Transmission Control Protocol) [4]. V rámci tohoto protokolu dochází k navázání spojení technikou „3-way handshake“, jak je vidět na obrázku 1.2. Iniciátor spojení, pro potřeby tohoto popisu značený písmenem A, nejprve zašle příjemci, zde označovanému jako B, SYN paket. B poté pošle A SYN-ACK paket. Za běžných okolností subjekt A potvrdí příjem SYN-ACK paketu, čímž je spojení navázáno. Informace o každém jednotlivém spojení se ukládají do TCB (Transmission Control Block), přičemž tento blok dat, v závislosti na operačním systému, dosahuje velikostí od 280 B do více než 1300 B.

Pokud však probíhá útok, k potvrzení přijetí SYN-ACK systémem A nedojde, a to buď protože výchozí IP adresa paketu SYN je jiná, než skutečná adresa A, nebo systém A neodpoví. Vzniklá situace vede ke vzniku polootevřených spojení. Pokud by takových spojení bylo neomezeně mnoho, systém B by po čase tvořením TCB vyčerpal celou paměť a selhal. Aby k této situaci nedocházelo, k jednotlivým portům bývá přidružen parametr backlog, který omezuje počet souběžných, byť polootevřených, spojení.

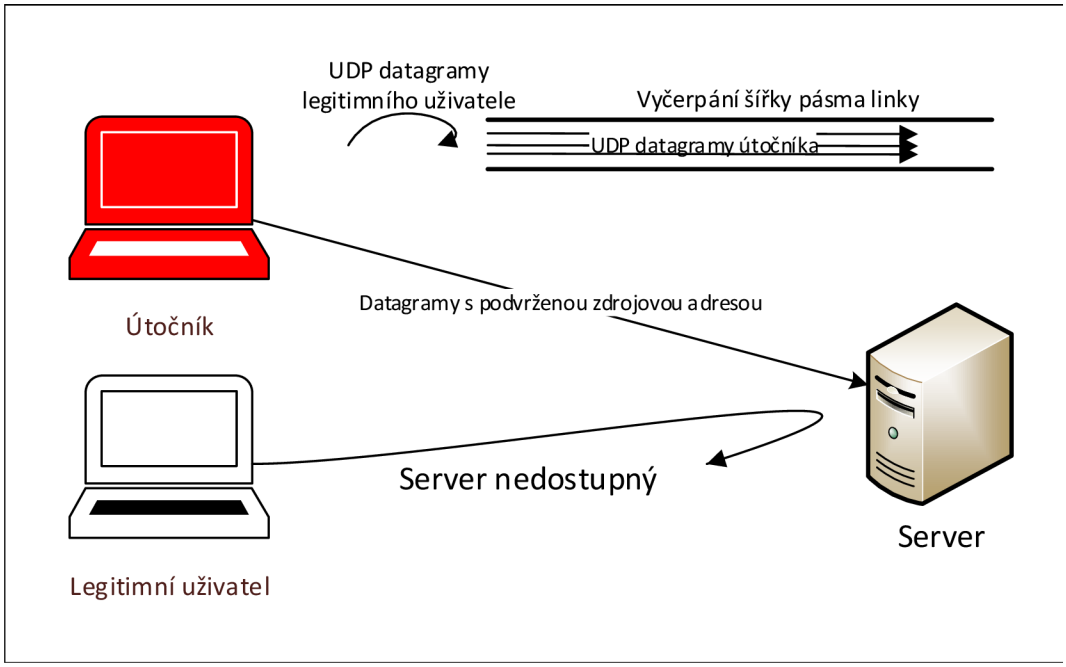


Obr. 1.3: Princip navazování polootevřených spojení během útoku SYN flood.

A právě vyčerpání počtu TCP spojení, a tím znemožnění vytváření dalších, je cílem tohoto útoku, protože tak dojde k odepření přístupu ke službě 1.3 legitimním uživatelům. Útok je v praxi různě modifikován, aby bylo jeho zachycení a filtrování obtížnější. Nejčastěji jsou podvržené zdrojové adresy paketu SYN nebo je ke generování paketů použit botnet. Díky principu vytváření nových polootevřených spojení, během kterých se stroj B opakovaně několikrát snaží odeslat SYN-ACK pa-

ket, než dojde ke smazání TCB, není pro dosažení efektu nutné útočníkem generovat velký objem dat. To ztěžuje rozpoznání útoku, neboť nedochází k takovému nárůstu síťového provozu, jako v případě jiných útoků.

Útok typu UDP Flood



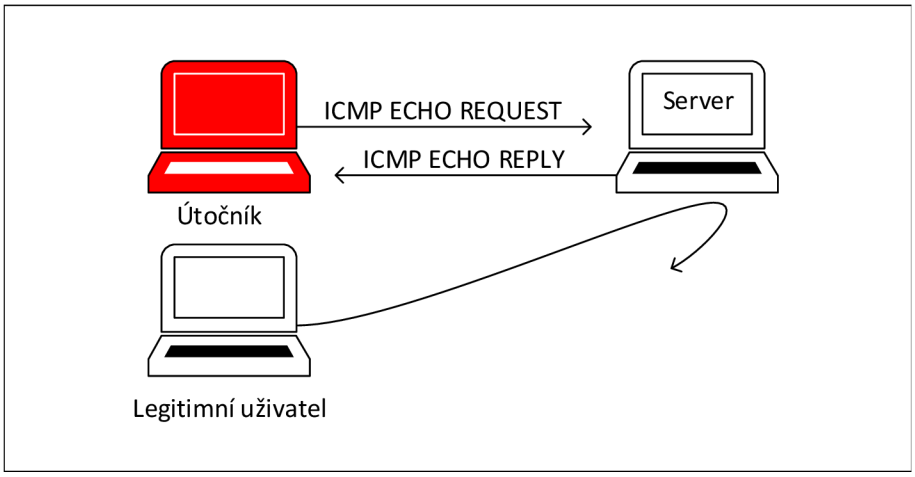
Obr. 1.4: Zahlcení serveru pakety s podvrženou zdrojovou adresou během UDP flood.

Protokol UDP (User Datagram Protocol) je nespojovaný, proto se na něj útočí hrubou silou generováním velkého objemu paketů [5]. To vede k přetížení serveru nebo síťových prvků na cestě k onomu serveru, jak je vidět na obrázku 1.4. Aby nebylo možné pakety z jednoho zdroje, útočníka, snadno odfiltrovat, používají se pro generované pakety podvržené zdrojové IP adresy nebo je provoz přímo generován distribuovaně, kupříkladu nějakým botnetem. V rámci UDP nenavazuje spojení, je obtížné rozeznat legitimní pakety od těch útočných. Filtrace obecně probíhá na základě dlouhodobější analýzy provozu na dané síti. Vizualizace vytvořená tímto projektem bude zajisté reflektovat masivní nárůst síťového provozu během tohoto útoku.

Útok typu ICMP Flood

ICMP je protokol používaný v TCP/IP sítích pro signalizaci chybových stavů [8]. Za normálních okolností jsou ICMP zprávy vytvářeny jednotlivými zařízeními v

síti. A to například ve chvílích, kdy není možné doručit pakety na cílovou IP adresu. Protokol není používaný v aplikacích pro koncové uživatele. Je nástrojem, který usnadňuje správu a diagnostiku sítě. Funguje nad protokoly IPv4 i IPv6.



Obr. 1.5: Zahlcení oběti během ICMP flood útoku.

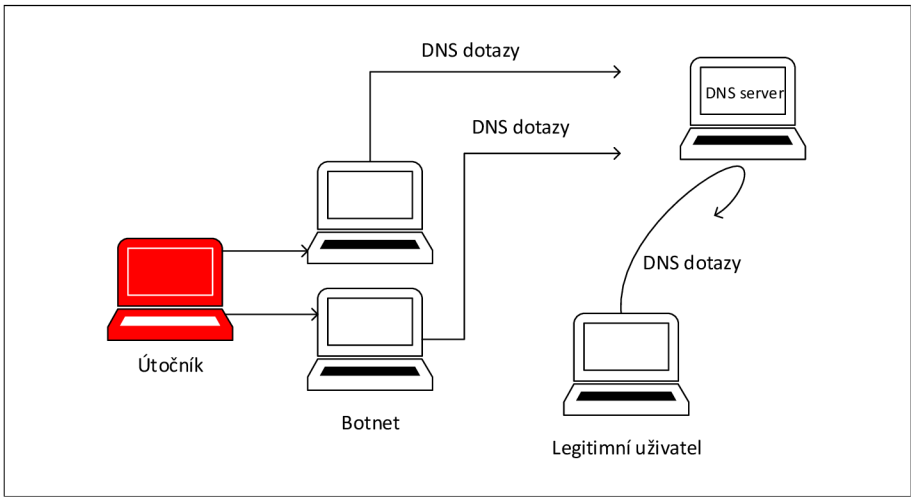
Jedním z příkladů použití ICMP je posílání echo-request paketů nástroje PING. Právě tyto pakety jsou v rámci ICMP útoku generovány útočníkem za účelem zahlcení serveru a tím způsobení jeho nedostupnosti, jak je znázorněno obrázkem 1.5. Dříve útočník generoval množství echo-request paketů s podvrženou zdrojovou adresou, aby se maskoval. V současnosti je k tomuto útoku používáno spíše botnetů napadených počítačů pod kontrolou útočníka, a proto není třeba zdrojové adresy modifikovat.

Prevencí takového útoku na odepření přístupu ke službě je monitorování množství ICMP požadavků a jejich případné blokování v případě jejich velkého množství.

Útok typu DNS Flood

DNS (Domain Name System) servery [11] zajišťují na Internetu překlad IP adres na doménová jména. Na jejich funkčnosti jsou závislí koncoví uživatelé, kteří by jinak museli pracovat přímo s IP adresami požadovaných webových stránek. DDoS útoky na ně nezpůsobí nedostupnost požadované stránky na internetu, ale znemožní její vyhledání prostřednictvím doménového jména 1.6. Tato situace může v uživateli bez odborných znalostí vyvolat dojem nedostupnosti samotné požadované webové stránky.

V současnosti je časté k realizaci daného útoku používání botnetů. Příkladem může být botnet Mirai [11]. Stejnojmenný malware napadá zařízení s operačním systémem Linux. Nemusí se jednat jen o počítače a servery. Upravené Linux distribuce jsou často provozovány na zařízeních, které je možné hromadně zařadit do



Obr. 1.6: Zahlcení oběti během DNS flood útoku.

skupiny IoT. Napadené stroje hromadně posílají DNS dotazy na DNS server. Ten je takovým provozem zahlcen a není schopen odpovídat legitimním uživatelům.

Ochrana proti takovým útokům je náročná. Problém je v odlišení relevantních dotazů a DNS dotazů útoku. Za tímto účelem se používají pokročilé detekční systémy, které sledují mnoho parametrů provozu.

Útok typu HTTP Flood

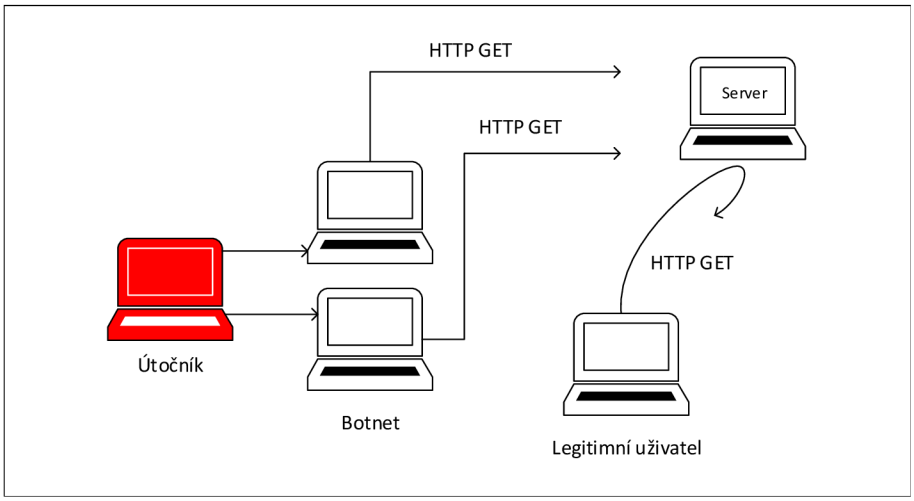
HTTP flood je typ DoS útoku, který působí nedostupnost serveru prostřednictvím posílání velkého množství HTTP GET či POST žádostí 1.7 na jeho IP adresu [28], čímž vyčerpá kapacity, určené legitimním uživatelům. Probíhá nad protokolem HTTP, případně HTTPS. Pokročilejší HTTP flood útoky mohou zvyšovat svoji efektivitu při snižování šířky pásma potřebné ke svému vlastnímu běhu například díky analýze výpočetní náročnosti odpovědí na HTTP GET dotazy. To umožní útočníkovi posílat velké množství HTTP GET takových, k jejichž zpracování potřebuje oběť velké množství výpočetních zdrojů.

Jako ochranu před takovým typem útoku slouží aplikační firewall, který pracuje na sedmé vrstvě síťového modelu ISO/OSI.

1.2 Současný stav vizualizace DoS a DDoS

Tato kapitola se zabývá hotovými vizualizačními nástroji, které jsou v současnosti dostupné pro grafické znázornění kybernetických hrozeb typu DoS. Dále obsahuje také analýzu technologií, jejichž použití je zvažováno pro účely této práce.

Hlavním důvodem vzniku potřeby vizualizovat útoky na odepření přístupu ke službě je jejich nárůst, například v minulých letech se jejich terčem staly velké známé



Obr. 1.7: Zahlcení oběti během HTTP flood útoku.

služby, jako Netflix, Twitter [25], který vyžaduje vývoj v oblasti protiopatřeních. Ta mohou být automatická nebo manuálně spouštěná síťovými administrátory. Jedno mají společné. Pro člověka je daleko přehlednější příslušné parametry, podle kterých se má nebezpečný provoz regulovat, vidět v nějaké grafické podobě, než pouze jako čísla. A to i v případě, že dochází k vizualizaci na nějaké páteřní síti s masivním provozem a z toho plynoucím obrovským množstvím zdrojové informace. V této práci bude pro lepší přehlednost vizualizace dělena na lokální v rámci nějaké sítě a globální.

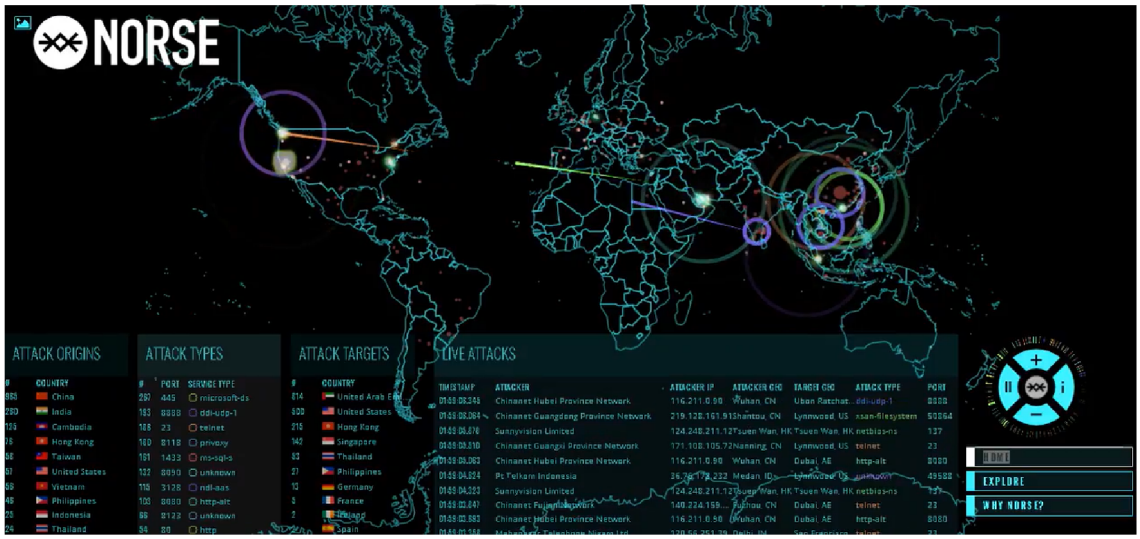
1.2.1 Vizualizace na úrovni sítě Internet

Do této kategorie se řadí zejména mapy kybernetických útoků. Jejich autoři sbírají data o síťovém provozu na páteřních sítích, ta pak graficky interpretují. Nyní uvedu několik nejznámějších kybernetických vizualizačních modelů [27]. Všechny mají společné propojení reálných geografických dat spolu se síťovým provozem.

Norse je jedna z nejznámějších map, umožňuje zobrazovat mnoho různých útoků, nejen DDoS. Zobrazení lze filtrovat na základě typů útoků nebo vyhodnocovaných síťových protokolů, odehrává se v reálném čase.

Uživatelské rozhraní 1.8 této webové aplikace se skládá z tabulek ve spodní části okna, ze kterých lze vyčíst informace o tom, kdo je oběť a kdo útočník, na jaké protokoly se útočí. Dále pak jaké porty jsou zneužívány. Zobrazené informace lze filtrovat dle různých kritérií. Asi nejzajímavějším prvkem je pak mapa celého světa, do které jsou zachycené aktivity vykreslovány.

Digital attack map je mapa vytvořená ve spolupráci s firmou Google, používá data z analytického systému ATLAS, vyvinutého firmou Arbor Networks, pro



Obr. 1.8: Mapa kybernetických útoků Norse.

detekci hrozeb a data několika stovek poskytovatelů služeb informační společnosti. Lze zobrazit informace o provozu jak v reálném čase, tak i v historii.

1.2.2 Vizualizace v rámci lokálních sítí

Do této kategorie lze zařadit jakýkoliv software, který tvoří grafické uživatelské rozhraní pro systémy IDS (Intrusion Detection System) detekující a IPS (Intrusion Prevention Systems) reagující na hrozby. Dále sem patří také programy pro analýzu síťového provozu na síťových rozhraních jednotlivých zařízení.

Scirius CE je software vytvořený pro potřeby ovládání systému IDS a IPS Suricata v grafické podobě. Je vyvíjen pod licencí GNU GPLv3. Jedná se o webovou aplikaci, primárně sloužící k ovládání systému IDS a IPS Suricata, avšak obsahující i grafické zobrazení statistik zachycených hrozeb.

Wireshark je asi nejznámější, multiplatformní, open source software, licencovaný jako GNU GPLv2, pro analýzu síťových protokolů [16]. Původním autorem je Gerald Combs, v současnosti do projektu přispívá celá řadou odborníků na počítačové sítě.

Program umí zachytávat veškerou komunikaci na síťových rozhraních zařízení a tu pak analyzovat. Nasbíraná data je možné filtrovat na základě mnoha kritérií, například dle použitého protokolu. Velkou výhodou je podpora mnoha protokolů Wiresharkem a z toho plynoucí možnost si pomocí jeho grafického rozhraní tyto protokoly rozebrat dle vnitřní struktury a zobrazit tak obsah přenášené informace. Pokud se jedná například o útok typu SYN flood, ve Wiresharku bude možné, za předpokladu vyfiltrování příslušných dat, přímo vidět nedokončené pokusy o navá-

zání TCP spojení. Navíc lze v rámci tohoto softwaru vytvářet grafy ze zachycených dat.

Oproti vizualizačním nástrojům z předchozí kapitoly, jejichž data bylo většinou možné interpretovat i bez hlubších odborných znalostí, je pro využití těchto nástrojů nezbytné určité povědomí o principech komunikace v počítačových sítích.

1.2.3 Možnosti využití jazyka JavaScript pro vizualizaci

JavaScript je objektově orientovaný programovací jazyk, vyvinutý společností Netscape, používaný pro tvoření dynamického obsahu na webových stránkách. HTML dokument, který tvoří obsah webové stránky, je totiž statický. Existují i různé alternativy, například VBScrtipt od společnosti Microsoft. V současnosti je však JavaScript nejpoužívanější.

Jedná se o interpretovaný programovací jazyk, to znamená, že je překládán do strojového jazyka až za běhu. Z toho plyne nižší rychlost vykonávání příkazů, oproti kompilovaným programovacím jazykům. Tato nevýhoda je z velké míry eliminována dobrou optimalizací v moderních webových prohlížečích.

V rámci této práce bude JavaScript použit jednak pro tvorbu interaktivních prvků na vytvořené webové stránce, jednak pro generování grafů z naměřených parametrů z filtračních zařízení. Pro potřeby tvorby grafů bude využito knihoven jazyka JavaScript. V následujících odstavcích provedu analýzu možností použití některé z knihoven pro vykreslování grafů v rámci projektu. Dále budou rozebrány možnosti využití některých vlastností technologie HTML5 za účelem generování grafiky pomocí JavaScriptu.

Canvas je HTML objekt, který slouží jako plocha pro vykreslování grafiky pomocí skriptů [18]. Může být použit pro vykreslování grafů. Je podporován ve všech moderních prohlížečích.

Chart.js je knihovna jazyka JavaScript [19], která využívá HTML objektu canvas pro vykreslování grafů. Zdroj [19] říká, že její využití naprosto minimálně ovlivňuje rychlost načítání stránky. Její velikost je možné zredukovat na 11 kb. Další výhodnou vlastností je schopnost přizpůsobit rozměry vykreslovaného grafu velikosti stránky. Výhodou je samozřejmě i velké množství dokumentace dostupné na webu a v neposlední řadě i fakt, že se jedná o open source projekt pod licencí MIT, tudíž by nebyl problém se zařazením knihovny do projektu z hlediska autorských práv.

Další knihovnou pro vytváření grafů pomocí JavaScriptu je **Canvas.js** vyvíjená indickou společností Fenopix. Opět používá HTML 5 objektu canvas pro vykreslování grafiky. Nabízí velké množství typů grafů a domovská stránka projektu slibuje

vysoký výkon i při vykreslování velkého objemu dat. Pro využití ve studentských projektech je poskytován zdarma.

Google Charts je open source knihovna [21], licencovaná pod Apache License 2.0. Nabízí velké množství různých grafů, vykreslovaných nad dynamickými daty, jejichž vzhled je navíc možné přizpůsobit. Autoři také slibují [21] podporu všech rozšířených prohlížečů a platforem. Grafy na webovou stránku knihovna vykresluje za pomocí vlastností technologie HTML5 a SVG. SVG je značkovací jazyk pro tvorbu 2D vektorové grafiky. HTML5 umožňuje vložení SVG kódu přímo do HTML kódu. Vzhledem k faktu, že za tímto projektem stojí společnost Google, je velice pravděpodobná jeho podpora i v budoucnu.

Flot, open source projekt pod MIT License. Zakladatelem je dánský vývojář Ole Laursen ze studia IOLA, které se zabývá vývojem pro web. Ke svému fungování používá další známé knihovny, jQuery. Jako prvek, na který se na stránce vykreslí graf, je použit prvek DOM (Document Object Model). Jedná se o jakési rozhraní, pomocí kterého mění JavaScriptový kód HTML a CSS obsah webové stránky a tím ji činí dynamickou. Výhodou této knihovny je podpora starších verzí prohlížečů.

1.3 Node.js

V rámci bakalářské práce je pro tvorbu specializovaného webového [12] serveru použita technologie Node.js. Jedná se o open-source systém pro tvorbu serverové části webových aplikací v jazyce JavaScript. Původním autorem projektu Node.js je Ryan Dahl. K interpretaci jazyka JavaScript v rámci Node.js je použit V8 JavaScript engine od společnosti Google. To je velmi výhodné, protože projekty vytvořené touto technologií disponují vysokým [12] výkonem, který plyne mimo jiné i z použití velmi dobře optimalizovaného interpreta z dílny společnosti Google [12]. Kód v jazyce javaScript je zpracováván v reálném čase.

Mezi velké a známé projekty, které používají Node.js patří například služba Netflix [9] pro streamování videa. Mezi hlavní důvody použití dané technologie, vývojáři zmíněné služby, patří škálovatelnost Node.js aplikací.

Další službou, která používá systém Node.js pro tvorbu backendu webové aplikace, je pracovní sociální síť LinkedIn. Dle zdroje [12] zde slouží Node.js server jako webová služba pro mobilní aplikaci. Vývojáři této služby zaznamenali mimo jiné i zvýšení výkonu, oproti stavu, kdy byla dané služba vytvořena pomocí jazyka Ruby. Dalším pozitivem byl rychlý vývoj služby díky jednoduchosti zápisu některých konstrukcí v jazyce JavaScript v rámci Node.js.

Klíčové vlastnosti technologie Node.js

Hlavním důvodem, který vedl k rozhodnutí vytvořit serverovou část tohoto projektu pomocí Node.js, byla skutečnost, že se v rámci této technologie aplikace vytváří v jazyce JavaScript, který je používán i v rámci prohlížeče klienta. Daný fakt bude usnadňovat práci na tomto projektu, který se skládá z více komponent, přičemž je v jeho podstatné části použito právě javaScriptu. K rozhodnutí využít Node.js vedl i fakt, že se jedná o aktuálně hojně používanou technologii pro webové služby v reálném čase i v rámci velkých projektů [12]. V neposlední řadě je třeba zmínit i dobrou dostupnost dokumentace Node.js.

Princip fungování Node.js aplikací

Aplikace Node.js z pohledu operačního systému serveru fungují jako jeden proces [12], který nevytváří nová vlákna pro každý požadavek ze strany webového prohlížeče klienta [12]. Zakládají se na použití neblokujících asynchronních funkcí. Díky tomu jsou velmi výkonné z hlediska obsluhy velkého množství klientů či požadavků. Pokud je nutné čekat na dokončení nějaké operace, například příchozí komunikace, jsou v daném okamžiku vykonávány jiné operace a přerušená operace je obnovena až v případě dostupnosti všech potřebných dat. O tento systém provádění jednotlivých operací se stará *Event Loop* systému Node.js.

1.4 Šablony pro tvorbu rozhraní

Tvorba přehledné webové stránky s účelným uživatelským rozhraním je komplikovaný problém. Je při ní třeba zvažovat rozložení ovládacích prvků, obsahu, volbu barev a fontů písma. Dále případně různé animace při dynamické změně obsahu.

Vzhledem k tomu, že množství webových stránek na Internetu neustále roste, a výše popsaným procesem již prošlo mnoho webových vývojářů, existují již hotové šablony, skládající se většinou jak z HTML dokumentu, tak i z připojeného CSS dokumentu. Část z nich je poskytována komerčně, avšak mnohé kvalitní jsou dostupné zdarma. Proto jsou pro účely tohoto projektu některé z nich představeny a budou diskutovány možnosti jejich použití.

Colorlib. je webová stránka, která se zabývá prodejem šablon pro webové stránky. Nabízí jednak placená komplexní řešení, jednak HTML šablony zdarma, často pod licencí CC BY 3.0, z toho vyplývá, že je na výsledném projektu nutné nechat jméno autora šablony. Avšak i tyto šablony je možné koupit a získat je tak pod jinou licencí bez nutnosti zmiňovat původního autora.

Free CSS.com opět nabízí jak řešení zdarma, tak i prémiové placené produkty. Nacházejí se zde šablony pro různé účely. Zajímavá je zde však nabídka již hotových komponent. Například navigační menu tvořené pomocí CSS a také rozložení obsahu

na webové stránce vytvořené styly CSS. Šablona v takové podobě by mohla být pro účely této práce vhodná, protože by usnadnila tvorbu kostry webové stránky.

Pro účely této práce by nebyl problém použít nějakou vhodnou šablonu pod licencí CC BY 3.0, avšak ta by většinou musela projít ještě velkými úpravami, a tak bude přistoupeno k tvorbě webové stránky zcela od začátku, avšak s možností využití dílčích, volně dostupných komponent.

1.5 Metody ochrany proti DoS a DDoS útoků

Pro subjekty ISP, ve smyslu provozování infrastruktury sítě Internet, a to ať jako poskytovatelů připojení, tak i různých služeb s touto problematikou spojených, je dostupnost sítí a zařízení v nich umístěných existenční otázkou. Proto jsou vyvíjeny různé metody, jak zabránit odepření přístupu útočníkem pro legitimní uživatele. V rámci této kapitoly bude rozebrána právě problematika obrany vůči těmto hrozbám.

1.5.1 Obecné přístupy k obraně proti DoS a DDoS útoků

Zdroj [1] rozlišuje čtyři obecné metody obrany proti útoku na odepření přístupu ke službě. Kritériem je zde umístění protiopatření.

Obrana na straně zdroje

Zachycení útoku u zdroje je teoreticky nejefektivnější, škodlivým provozem totiž nejsou zatěžovány další sítě a v nich probíhající legitimní provoz. Navíc takový systém vyhodnocuje malé množství síťového provozu, a tak není ani příliš výpočetně náročný. Hlavními slabinami této techniky je obtížná detekce v případě distribuovaného útoku. Ta je způsobena velkým množstvím zdrojových stanic, kdy každá jednotlivá generuje provoz, objemem blížícím se běžnému provozu.

Příkladem systému fungujícího na zmíněném principu je projekt D-WARD [10]. Ten používá analytický software, běžící na jednotlivých routerech přímo v koncových lokálních sítích a detekuje odchylky od schématu běžného provozu, které pak filtruje.

Obrana na straně cíle

Při použití tohoto způsobu obrany se předpokládá využití síťových prvků v síti cíle k eliminaci hrozby. Takový systém může efektivně vyhodnocovat provoz a jeho odchylky v reálném čase. Další výhodou je autonomnost, kdy je systém obrany nezávislý na okolních sítích a nemusí spoléhat na obranu ze strany jiného subjektu. Tento přístup má i svá úskalí, jsou kladeny velké nároky na výkon filtračních zařízení.

Dalším problémem je fakt, že útoky, jako je třeba UDP flood, mohou být úspěšně odfiltrovány, avšak pakety jimi generované i přesto využijí kompletní šířku pásma linky vedoucí právě k filtračním subjektům.

Obrana na úrovni páteřních sítí

Použití tohoto schématu obrany vyvažuje kompromis mezi dvěma hlavní parametry, které jsou sledovány odděleně v případě obrany na straně cíle, nebo útočníka. Jednak přesnost detekce hrozby a pak také šířku pásma, využitou k útoku.

Problematika úzce souvisí s distribuovanou obranou, která bude podrobněji probrána v následující podkapitole. Avšak asi nejdůležitější je zde fakt, že v tomto případě je provozovatel obranných mechanismů současně správcem nějakého většího síťového celku, a proto je schopen v rámci tohoto síťového celku provozovat síťové prvky, schopné škodlivý provoz detekovat a odfiltrovat. Přidanou hodnotou je schopnost komunikace mezi těmito prvky, což zvyšuje efektivitu vlastního přístupu k ochraně.

Distribuovaná obrana

Jistá forma distribuovanosti, ve smyslu komunikace a propojení jednotlivých obranných mechanismů, je obsažena ve všech předchozích popsaných filosofiích. Obecně se zde předpokládá přítomnost mechanismu pro filtraci škodlivého provozu na každém aktivním síťovém prvku, jakým je například router. Achilovou patou systému je obtížná kontrola, zda je zmíněný předpoklad platný. K nefunkčnosti totiž stačí absence nějakého ochranného systému i na pouze několika zařízeních.

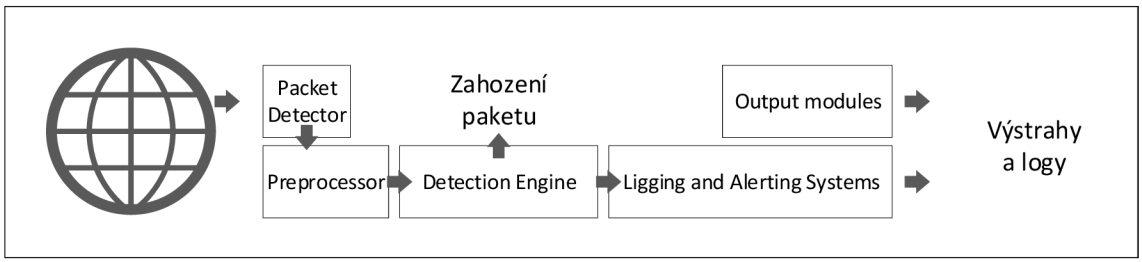
Dalším příkladem distribuované ochrany před útoky na odepření přístupu ke službě je cloudový přístup. Firmy, které se zabývají informační bezpečností, nabízí řešení, kdy se k očištění legitimního provozu od provozu generované útokem využívá jejich infrastruktura. Výhodou je jednoduché zavedení takového systému k zákazníkovi, kdy stačí jeho infrastrukturu pouze zapojit do již hotového systému. Dalším pozitivem je velká efektivita, která plyne ze zkušeností firem, jako například Kaspersky [13], s obranou vůči DDoS. V rámci těchto systémů je u zákazníka pouze sonda k detekci škodlivého provozu [13], komunikace mezi serverem zákazníka a uživateli probíhá přímo. Pouze během útoku je vedena přes zařízení k čištění provozu společnosti Kaspersky.

1.5.2 Vybrané systémy DoS a DDoS obrany

V rámci podkapitoly budou představena některá hardwarová a softwarová řešení obrany proti útokům na odepření přístupu ke službě.

Software na filtraci DoS a DDoS

Snort je open source systém pro analýzu síťového provozu v reálném čase. Autorem je Martin Roesch. Program pro své fungování využívá libpcap. Knihovna libpcap je určena pro zachytávání dat ze sítě v reálném čase. Jedná se o implementaci pro unixové systémy.



Obr. 1.9: Schéma IPS Snort.

Během svého fungování systém používá pravidla detekce anomálií i analýzu protokolů. Skládá se z pěti základních komponent, jak je vidět na obrázku 1.9. Packet Detector slouží k uložení paketů ze síťových rozhraní a jejich následné přípravě ke zpracování. Preprocessor rozpoznává anomálie v hlavičkách paketů. Dalším modulem je Detection engine, zde se aplikují pravidla, zadaná do systému Snort, na jednotlivé pakety. Poslední dvě komponenty fungují jako výstupy z aplikace.

Systém Snort je velmi používaný, jeho hlavní nevýhodou je pouze jednovláknové zpracování paketů a z toho plynoucí nižší rychlost. Další možností, jak ochránit síť, je systém **SURICATA**. Opět se jedná o open source software, používaný však často i v komerčních řešeních. Je schopný fungovat jako IDS i IPS systém v reálném čase. Pro svůj běh využívá rozsáhlá pravidla a signatury. Veliká výhoda SURICATY oproti Snortu je rychlost, který je zapříčiněna vícevláknovým zpracováváním provozu.

Hardwarová řesení pro filtraci DoS a DDoS

V tomto případě se jedná o komplexní řešení, kdy dodavatel poskytne zařízení, fyzicky optimalizované pro funkci IPS. Společnost **McAfee** nabízí například McAfee Network Security Platform 1.10. Takové přístroje využívají proprietární systém, předkonfigurovaný pro potřeby jednotlivých zákazníků.

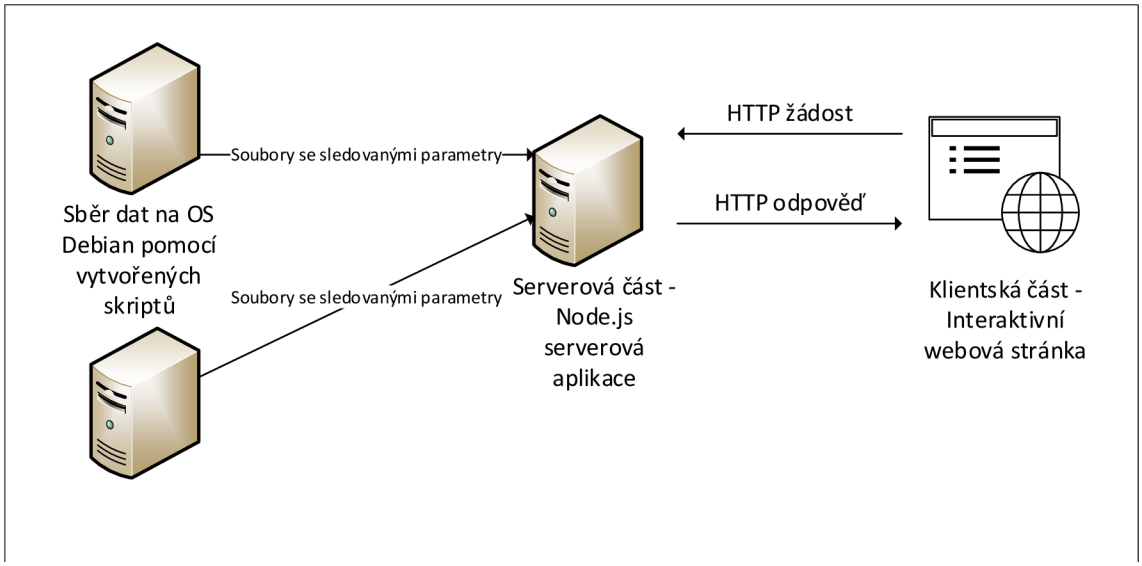
Obr. 1.10: McAfee Network Security Platform.

2 Praktická část bakalářské práce

Předchozí kapitoly obsahovaly analýzu nástrojů, jejichž užití je pro tuto práci zvažováno, definici parametrů, které které budou sledovány v rámci tohoto projektu. Tato kapitola se věnuje praktickému projektu, založenému na technologiích, vybraných na základě analýzy v předchozích kapitolách. Obsahuje popis vlastního návrhu a implementace vytvořené softwarové aplikace, použitých knihoven, nástrojů a také seznámení se školním experimentálním pracovištěm. Součástí je testování implementace na školním experimentálním pracovišti.

2.1 Interaktivní webová stránka

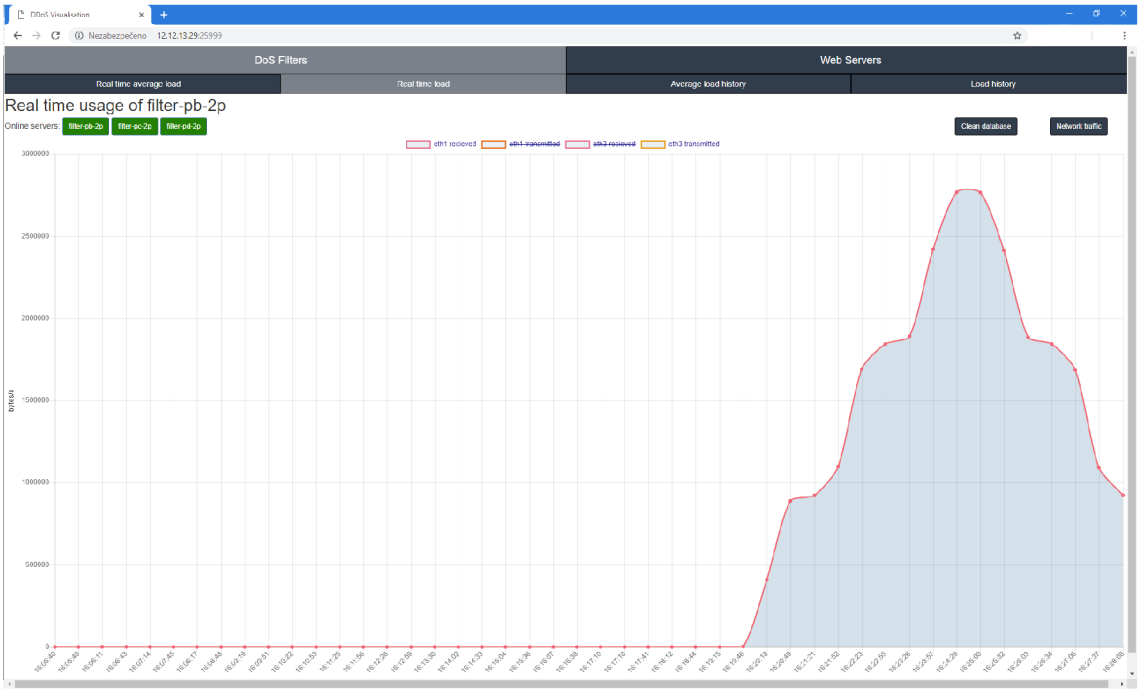
Celé technické řešení praktického projektu bakalářské práce se skládá z několika částí, které spolu spolupracují na vytvoření výsledné uživatelské zkušenosti. Návrh základní architektury systému vizualizace DDoS je vidět na obrázku 2.1. Zobrazená webová stránka tvoří klientskou část řešení, určenou k prezentaci naměřených hodnot v grafické podobě. Další součástí je serverová část projektu. Jedná se o serverovou aplikaci, která má na starosti zpracovávat naměřená data a ta pak poskytovat klientské části v definovaném formátu k dalšímu použití. Jejím dalším úkolem je poskytování potřebných souborů webové stránky internetovému prohlížeči klienta. O samotný sběr dat ze sledovaných serverů na experimentálním pracovišti VUT a následný transport těchto dat na stroj, na kterém běží serverová část projektu, se starají skripty. Jejich bližšímu popisu je věnována část této kapitoly.



Obr. 2.1: Schéma technického řešení bakalářské práce.

Klientská část

Je tvořena webovou stránkou s grafickým uživatelským rozhraním pro sledování dopadů DDoS útoků na experimentální pracoviště. Ta má dle zadání umožnit uživateli interakci se svým obsahem. Obsahuje proto tlačítka a jiné aktivní prvky, umožňující uživateli přizpůsobit aktuální zobrazení dat svým potřebám. Stránka byla vytvořena jako *single page*, to znamená, že veškerý obsah je na ní dynamicky měněn pomocí JavaScript kódu a během používání stránky tak nikdy nedochází k jejímu kompletnímu načítání, což má pozitivní vliv na výslednou uživatelskou zkušenost.



Obr. 2.2: Rozhraní vytvořené interaktivní webové stránky.

Z pohledu grafického uživatelského rozhraní je zhotovená stránka rozdělena, jak je vidět na obr. 2.2, na navigační menu v horní části a prostor s vykresleným grafem, který se nachází v oblasti pod menu a vyplňuje většinu prostoru okna prohlížeče.

Navigační menu se skládá z tlačítek pro volbu požadované oblasti ke sledování, informativního výpisu, řádku s indikátory online serverů a tlačítka pro mazání dat z logů či tlačítka pro přepnutí mezi vykreslením dat o síťovém provozu a využitím výpočetních prostředků sledovaného zařízení.

Dvě velká tlačítka v prvním řádku slouží k přepínání mezi sledováním využití filtračních a webových serverů. Menší tlačítka v druhém řádku plní roli přepínačů typu grafu k vykreslení. Jsou zde čtyři možnosti. Uživateli je nabízena volba mezi zobrazením dat v reálném čase či dat historicky zaznamenaných. Dále pak mezi zobrazením dat pro konkrétní sledovaný server, nebo dat agregovaných ze všech

zařízení z dané kategorie. Zda tato čtyři tlačítka zobrazují grafy vytvořené na základě dat ze sledování filtračních či webových serverů záleží na aktivní volbě sledované oblasti v prvním řádku menu. V závislosti na tom, jaké volby jsou aktuálně aktivní, je měněn status a barva tlačítek menu.

Pomyslný třetí řádek v rozložení grafického rozhraní vyplňuje informativní výpis. Jeho účelem je poskytnout uživateli informaci o aktuální pozici v rámci webové stránky a aktuálně vykreslovaném grafu.

Poslední část menu se skládá z indikace online serverů. Ta má formu tlačítek, která mění barvu v závislosti na skutečnosti, zda je daný server online. Pokud se uživatel nachází v oblasti, která má na starosti zobrazování dat z jednoho konkrétního serveru, mají tato tlačítka funkci voliče takového serveru. V ostatních případech jsou neaktivní a mají pouze informativní funkci. Ovládací prvek pro mazání historických dat promaže veškeré soubory logů na webovém serveru. Přepínač mezi zobrazením využití výpočetní kapacity a síťového rozhraní sledovaného zařízení reprezentuje tlačítko v pravé části tohoto řádku.

Samotný graf obsahuje popisky os a ovládací prvky k zapnutí či vypnutí zobrazení vybraných dat.

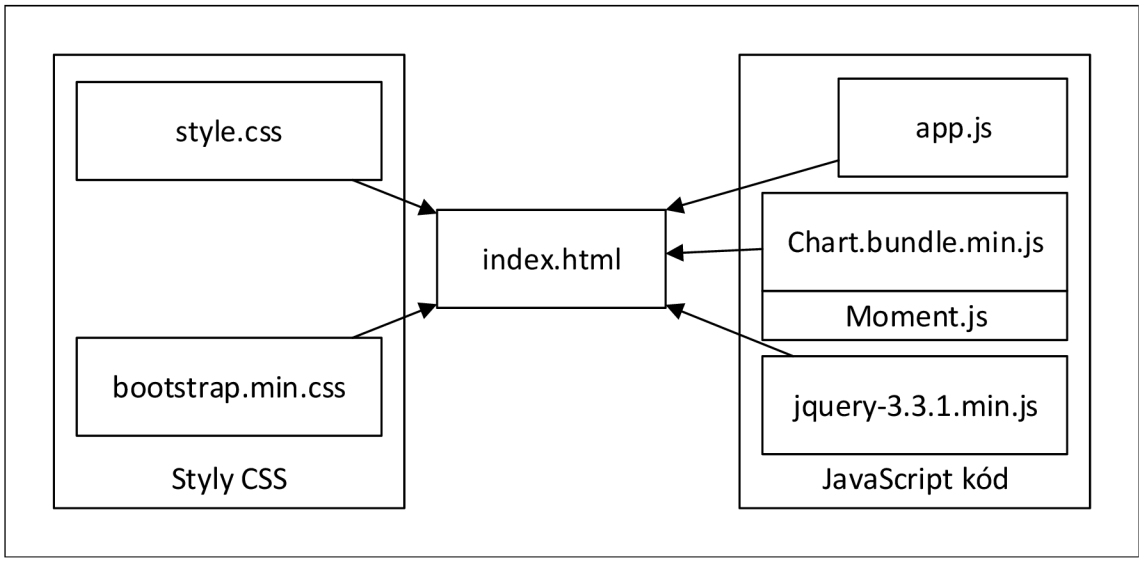
V případě, kdy graf pro své vykreslení, například dat z historie, potřebuje ze strany uživatele zadat časové intervaly, zobrazí se pod ním textové pole s předdefinovaným formátem pro zadání časového údaje. Tento formát je třeba přesně dodržet pro správnou funkčnost. Pokud se uživatel nachází v oblasti vykreslování jednotlivých konkrétních grafů, graf je vykreslen po stisknutí tlačítka specifického serveru z řádku indikace online serverů. V případě, že se jedná o oblast vykreslování grafů z dat agregovaných, nachází se vedle textových polí pro vložení informace o časovém intervalu ještě tlačítko Plot pro vykreslení grafu.

Z hlediska principu fungování je web vystavěn na třech technologiích. HTML dokumentu, který tvoří strukturu a obsah stránky, CSS souborů definujících grafický styl a zejména pak funkční části, napsané v jazyce JavaScript. Schéma zdrojových kódů samotné webové stránky je na obr. 2.3.

Použité knihovny

Chart.js je open source knihovna jazyka JavaScript. Teoretické aspekty byly probrány v kapitole 2.3, zde jsou zmíněny klíčové vlastnosti, kvůli kterým byla knihovna použita.

V její prospěch hrála dostupnost dokumentace, ale i různých návodů. Tento fakt psaní webové stránky zjednodušil. Výhodou je velká variabilita ve způsobech vykreslení os grafu. Dalším pozitivem je integraci schopnosti přizpůsobit obsah vykreslovaného grafu velikosti stránky. Dále prvky, jako jsou interaktivní popisky os grafů,



Obr. 2.3: Schéma kódu interaktivní webové stránky.

které reagují, pokud je na ně najeto kurzorem myši. Praktická je i možnost vypnout dočasně zobrazení některé z vykreslených křivek grafu.

Výhodný je i výkon takového řešení. Během testování nedocházelo ke zpomalování, ani pokud graf obsahoval v řádu desítky uzlů. Samozřejmě vzhledem k rozlišení zobrazovacího zařízení nedává smysl ani z grafického, ani z výkonnostního hlediska vykreslovat řádově stovky až tisíce bodů, a proto je maximální počet vykreslených bodů v rámci kódu omezen. K některým funkcím v oblasti práce s datem a časem potřebuje tato knihovna open source knihovnu **moment.js**. Ta je dostupná pod licencí MIT.

jQuery, knihovna jazyka JavaScript s otevřeným kódem [20]. Chráněna svobodnou MIT licencí. Charakteristické vlastnosti jsou rychlost, malá velikost, multiplatformní podpora většiny prohlížečů. Zlepšuje a zjednodušuje spolupráci mezi JavaScriptem a HTML dokumentem. Například obsahuje funkce, které výrazně zkrátí psaní jinak dlouhých konstrukcí, třeba jako AJAX (Asynchronous JavaScript and XML).

Bootstrap byl vyvinut a vydán v roce 2012. Je to open source CSS framework, pod licencí MIT. Autoři jsou Mark Otto a Jacob Thornton. Původně se jednalo o interní nástroj společnosti Twitter. Pro projekt je užito funkce rozdělení na kontejnery a řady pro strukturování obsahu a jeho rozložení na stránce. Díky tomu je výsledný web kompletně responzivní. To znamená, že své zobrazení přizpůsobuje rozlišení okna, na které je vykreslen. V rámci práce je použito také předdefinovaných vzhledů tlačítek a jiných ovládacích prvků. Díky tomu design stránky působí konzistentně.

Klíčové části zdrojového kódu vytvořené webové stránky

Soubor, který odkazuje na veškeré ostatní části vytvořené webové stránky projektu bakalářské práce je **index.html**. Zobrazí se při načtení stránky prohlížečem a na jeho objektech dochází ke generování grafického rozhraní. Sestává se z hlavičky a těla.

Hlavička obsahuje pouze odkazy pro načtení souborů se styly. Dále pak titulek. Ten se zobrazí na kartě okna ve webovém prohlížeči.

Tělo se pak skládá z kontejneru, řad a jednotlivých sloupců, umístěných do řad. Hierarchie členění vychází z principů rozložení knihovny Bootstrap popsané výše. Ve sloupcích se nachází funkční tlačítka pro ovládání aplikace a další objekty, jako například textová pole či canvas objekt. Ten byl popsán již v teoretické části. Jedná se o plochu pro zobrazení grafiky, v tomto případě plní canvas roli pomyslného plátna pro vykreslení grafu. V těle stránky se nachází také odkazy na použité JavaScript soubory a knihovny.

style.css je dokument stylů, pomocí kterého je nakonfigurován vzhled stránky. Prostřednictvím style.css jsou definovány barvy tlačítek, pozadí, vzhledy tlačítek a jiných objektů. Dále také prostřednictvím modifikací tohoto dokumentu dochází k ovládání zobrazení jednotlivých komponent v příslušný moment.

app.js je soubor s JavaScript kódem. V rámci něj je použito funkcí ze zmíněných knihoven. Text popisuje klíčové části vytvořeného zdrojového kódu, spouštěného v rámci vzniklé webové stránky prohlížečem klienta. Kompletní zdrojový kód, o kterém se zde hovoří, je součástí elektronické přílohy bakalářské práce.

V případě, že klient ve svém webovém prohlížeči načte stránku vytvořenou v rámci práce, je pokaždé načtena tato stránka v inicializačním stavu. To znamená, že zobrazuje agregovaný graf využití síťových rozhraní všech dostupných filtračních serverů. Uvedení stránky do tohoto stavu má na svědomí anonymní callback funkce, která je spuštěna příslušným listenerem, kontrolujícím načtení stránky. Zmíněná funkce zobrazí příslušné ovládací prvky spojené s inicializačním stavem. Do proměnné uloží textový řetězec, který označuje daný stav, v němž se stránka nachází. Dále provede asynchronní volání na server, aby provedla zjištění, které servery z experimentálního pracoviště jsou dostupné ke sledování a dle toho vygenerovala indikační tlačítka s jejich názvy. Poté spustí `NetAverage()`. Daná funkce se stará o vykreslení příslušného grafu, jejímu popisu se bude text věnovat později. Posledním krokem je spuštění časovače, periodicky kontrolujícího status dostupných serverů. V rámci tohoto časovače je na základě faktu, zda jsou sledované servery online, měněna barva tlačítek s jejich názvy.

Funkce `checkStatus()` existuje ve verzi pro sledování statusu filtračních i cílových serverů. Pokaždé funguje obdobně, nejprve modifikuje styl barvy tlačítka na

červenou pro všechna tlačítka. Poté provede příslušné volání na server, který jí vrátí JSON objekt, obsahující názvy online serverů z požadované kategorie. Na základě získaných dat jsou dané ovládací prvky, s logy jednotlivých aktuálně dostupných serverů, obarveny do zelené barvy.

Každý graf vytvořený pomocí knihovny Chart.js je reprezentován objektem, obsahujícím definice podoby os, barev a stylů zobrazení. V neposlední řadě i polem hodnot souřadnic jednotlivých záznamů. Zobrazování samotných grafů mají na starosti v případě, že se jedná o jedno z možných zobrazení v reálném čase, funkce jako například `NetAverage()`, zmíněná v jednom z předchozích odstavců. Každá z těchto funkcí má, pokud je to nutné, podmínku, pomocí které dokáže graf přizpůsobit jak oblasti sledovaní filtračních, tak i cílových serverů.

Tělo každé z těchto funkcí obsahuje následující sekvenci. V první části dojde k vymazání všech dříve vzniklých objektů grafu, které už nejsou v daném okamžiku potřeba. Následuje volání inicializační funkce pro vznik nového grafu. Poté je vytvořen časovač s časovým intervalem definovaným pomocí globální proměnné. V rámci časovače dochází v každém intervalu k volání na server, který odpovídá potřebnými daty. Ta jsou rozparsována a pomocí příslušné funkce, určené k přidávání hodnot do datových struktur typu array, sloužících jako zdroj dat pro objekt grafu, přidaná právě do oné datové struktury aktuálně vykreslovaného grafu. Příkladem funkce zmíněné v předchozí větě je třeba `addValuesNetwork1Interface()`. Všechny funkce tohoto typu jsou volány s parametrem, kterým je číslo omezující počet prvků v zdrojových datových strukturách objektů grafů. Tím je ošetřeno množství vykreslených hodnot. Posledním krokem je aktualizace vykresleného objektu grafu.

Pokud uživatel zvolí některý z grafů, jež pracuje s historickými hodnotami, je postup obdobný, jako v předchozím případě, avšak s jednou zásadní změnou. Volána je zde například `DatabaseNet()` s parametry určujícími časové okno k vykreslení. Tentokrát proběhne pouze jedno volání na server, doplněné o časové hodnoty. Server odpoví JSON objektem. Časovač je zde nahrazen for cyklem. Každý krok je podobný kroku v rámci časovače. JSON objekt je zpracován jako pole jednotlivých záznamů. V každém kroku je procházen jeden záznam, ten je parsován a zpracován ke zobrazení. K updatu grafu dojde až po projití a zpracování celého pole záznamů.

Důležitou roli hraje `deleteAll()`. Stará se o mazání nepotřebných objektů, nulování obsahů globálních proměnných, časovačů. Je volaná před každým vytvořením nového grafu, aby nedocházelo k chybám způsobeným kolizí dat či přepisováním existujících objektů grafů. V rámci projektu jsou vytvořeny globální proměnné, do kterých se ukládají informace o aktuální poloze uživatele v rámci grafického rozhraní. Díky nim je možné upravovat chování jednotlivých funkcí na míru požadovanému zobrazení grafu. Všechny funkce jsou vhodně propojeny do akcí stisknutí jednotlivých tlačítek.

Serverová část

Aplikace, která slouží jako webový server pro vizualizační webovou stránku, vznikla na míru tomuto projektu. Je napsaná v jazyce JavaScript pomocí technologie Node.js. Jedná se o prostředí pro tvorbu serverových aplikací v JavaScriptu. Je multiplatformní, rychlé, efektivní a dobře dokumentované.

V rámci Node.js jsou používány moduly. O tématice hovoří zdroj [7]. Jedná se o obdobu knihoven. Moduly jsou soubory, které obsahují zdrojové kódy hotových funkcí. Je možné jednak používat moduly třetích stran či vlastní moduly, jednak lze využít moduly vestavěné. Ty není nutné instalovat.

Pokud je potřeba použít v aplikaci funkci z nějakého modulu, je nutné nejprve daný modul klíčovým slovem `require()` importovat a přiřadit mu proměnnou. S daným modulem se pracuje jako s objektem. V rámci bakalářské práce je použito několik vestavěných modulů. Hlavními jsou modul **http** pro vytvoření webového serveru, **fs** pro práci se souborovým systémem a **child_process** pro spouštění jiných programů pomocí Node.js.

Webový server

Každá serverová aplikace, vytvořená v Node.js, musí v určitém bodě obsahovat tvorbu objektu webového serveru [6]. Stejně tomu je i v rámci tohoto projektu. Vytvoření serveru provádí funkce `createServer()`, volaná nad proměnnou, do které je importován http modul. Jejím Argumentem je definice anonymní funkce, jež je prováděna s každým HTTP voláním na server.

Strukturou anonymní funkce je podmínka. Ta tvoří rozhraní pro jednotlivá HTTP volání. Vždy je vyhodnocena URL adresa HTTP žádosti. Na základě jejího obsahu podmínka rozhoduje, jaké akce budou vykonány. Například pokud URL obsahuje pouze znak /, což je kořen adresáře, je klientovi vrácen dokument index.html, na který jsou navázány veškeré další soubory, potřebné pro běh webové stránky grafického rozhraní.

Všechny možnosti, ve které může podmínka vyústit, tvoří pomyslné cesty pro zpracování příchozích požadavků od klientů. Jejich obecná struktura je pokaždé podobná. Vždy se skládají z přijaté HTTP GET žádosti a vytvoření odpovědi na tuto žádost. Každá odpověď serverové aplikace sestává z hlavičky a těla. Hlavička v případě, že nedojde k chybovému stavu, obsahuje HTTP stavový kód 200, který značí, že je vše v pořádku. Pokud ale nastane situace, kdy pro URL neexistuje žádná definované cesta pro zpracování, klientovi je vrácena odpověď s chybovým kódem, například 404. Součástí hlavičky je informace o typu dat, přenášených v těle zprávy. Tělo zprávy je tvořeno buď požadovaným dokumentem, nebo daty ve formátu texto-

vého řetězce. Pokud dochází k přenosu dat v nějaké složitější struktuře, jako je třeba pole, data jsou přenášena jako JSON objekt. Jedná se o obecný formát zápisu dat. Daty je vždy odpovídáno na asynchronní volání ze strany JavaScriptového kódu, který je vykonáván v rámci webového prohlížeče klienta jako součást interaktivní webové stránky.

Je možné nastavit port pro naslouchání vytvořeného webového serveru.

Zpracování záznamů

Ze strany sledovaných filtračních a webových serverů jsou data odesílána ve formě souborů s jednotlivými záznamy na stroj, na kterém běží aplikace webového serveru, vytvořeného v rámci bakalářské práce. V rámci dané Node.js serverové aplikace jsou tyto soubory čteny, zpracovávány a jejich obsah je pak v případě HTTP žádostí, ze strany klientů, předáván internetovým prohlížečům těchto klientů. Nyní budou rozebrány základní principy akcí, jež jsou vykonány po přijetí HTTP GET požadavků na určitou URL adresu.

Na žádosti **`/singleFileLive`** server odpovídá přečtením souboru logu s pouze jedním záznamem o využití konkrétního sledovaného serveru a předáním takto zjištěných dat webové stránce na straně klienta. Součástí URL požadavku je i název souboru logu konkrétního serveru.

V případě, že server obdrží volání ve tvaru **`/singleFileFromTo`**, je prohlížeči klienta odpovězeno JSON objektem, složeným z jednotlivých záznamů souboru logu konkrétního sledovaného serveru. Záznamy pokrývají vždy časový interval určený časovými značkami, jež jsou spolu s názvem požadovaného serveru součástí volané URL.

Principy zpracování **`/multiFileLive`**, **`multiFileDatabase`**, **`/multiFileWebLive`** a **`multiFileWebDatabase`** jsou obdobné, jako předchozí dva případy, avšak s několika zásadními rozdíly. Jednak je zde pokaždé zpracováváno současně více souborů logů. Dochází k průměrování dat o využití RAM a CPU. Počty přenesených bajtů na síťových rozhraních se vhodně sčítají. Jednak první dvě volání v tomto odstavci interagují s daty o využití filtračních serverů a druhá dvě pak s daty o využití cílových serverů.

Odpovědi na **`/onlineServers`** a **`/onlineWebServers`** tvoří seznamy všech aktuálně dostupných serverů. Kritériem, které určuje, zda je server dostupný, je aktuálnost posledního získaného záznamu v rámci logu z tohoto sledovaného serveru. Vytvořené funkce kontrolují rozdíly aktuálního času a časového razítka daného záznamu. Jako online je označen server, kde se rozdíl aktuálního času a časového razítka záznamu liší méně, než je stanovaná hranice dvou minut.

Seznam všech serverů, jejichž soubory záznamů se nacházejí na serveru s běžící webovou aplikací je dostupný pro cílové servery pod URL `/AllServersW` a pro filtrační pod URL `/AllServersD`.

V rámci volání `/cleanDATA` dojde k pročištění souborů logů. Jelikož jsou synchronizovány záznamy ve formě logů na webovém serveru se záznamy lokálně na jednotlivých sledovaných strojích, je nutné čištění provádět jak na úrovni souborů webového serveru, tak i vzdáleně na jednotlivých zaznamenávaných strojích. Serverová aplikace proto postupně vytváří v rámci tohoto volání skripty, které mají za úkol provést samotné mazání. Tyto skripty pak vzdáleně spouští na sledovaných strojích. Na konci dochází k odstranění souborů skriptů a veškerých logů na straně serveru.

Sběr provozních dat na OS Debian

Při návrhu způsobu sběru dat na operačním systému Debian, na kterém fungují filtrační a cílové servery, bylo třeba zvážit několik aspektů. Jednak která data budou sbírána pro následnou grafickou interpretaci uživateli, jednak jakým způsobem bude sběr probíhat.

Jako metrika pro zobrazení byla po konzultaci s vedoucím práce stanovena základní data o provozu počítače, ať už jím je fyzický či virtuální stroj. Jedná se o procentuální vytížení procesoru, operační paměti RAM a data o provozu na síťové kartě. Zde je zvoleno množství bajtů příchozích i odchozích z jednotlivých síťových rozhraní.

Když byla stanovena sledovaná metrika, bylo nutné navrhnout způsob jejich čtení a zpracování. Zde bylo využito možností bashe. Jedná se o shell UNIX systému pro interpretaci příkazového řádku. Obsahuje různé vestavěné příkazy, ale i příkazy podporované dalšími programy pro příkazovou řádku. Konkrétněji byl vytvořen skript. Je to malý program, využívající syntaxe příkazů podporovaných v příkazové řádce.

Skript má následující strukturu. Obsahuje nekonečný cyklus while, v rámci kterého vykonává periodicky v časovém intervalu dílčí příkazy.

Prvním krokem je čtení potřebných dat o počtu přenesených bajtů ze souboru /proc/net/dev. Čísla jsou vyfiltrována za pomocí příkazu AWK. AWK je kompletní programovací jazyk pro práci s textem. Je interpretován stejnojmenným programem. Přečtená data, která vyjadřují počet přenesených bajtů od spuštění systému, jsou následně uložena do proměnné. Celý proces se po třiceti vteřinách opakuje. Následně je ze získaných dat vypočten počet bajtů přenesených za vteřinu. Tato hodnota vyjadřuje průměrnou hodnotu za třicet vteřin.
V dalším kroku se využívá výpisu na konzoli, v rámci něj pak příkazu top, který slouží v UNIX systémech k monitorování běžících procesů, pro čtení aktuálního vy-

užití procesoru a paměti RAM. V rámci výpisu je opět použit příkaz AWK, dále pak příkazy jako grep a tail pro zpracování dat v textu. Celý výpis je ještě doplněn o data ze síťových rozhraní a přesměrován do dvou souborů log.txt a input.txt, které slouží jednak jako archiv všech naměřených hodnot, jednak jako zdroj poslední naměřené hodnoty. Názvy souborů log.txt a input.txt jsou doplněny o IP adresu servisního síťového rozhraní daného stroje a název tohoto stroje. Tyto další informace jsou použity v rámci zpracování dat serverovou aplikací.

Posledním problémem byl transport dat na serveru pro zpracování a předání stránce k vizualizaci. Úkol se podařilo vyřešit nástrojem rsync pro tvorbu záloh a vzdálenou synchronizaci souborů. Protokolem, pomocí kterého jsou data přenášena, je SSH (Secure Shell), to znamená, že jsou během transportu v šifrované podobě.

Skript obsahuje i pojistku proti vytváření příliš dlouhých logů s velkým množstvím hodnot. Data jsou každý den v definovaný čas promazána a jejich záznam probíhá znovu od začátku.

2.2 Popis experimentálního pracoviště

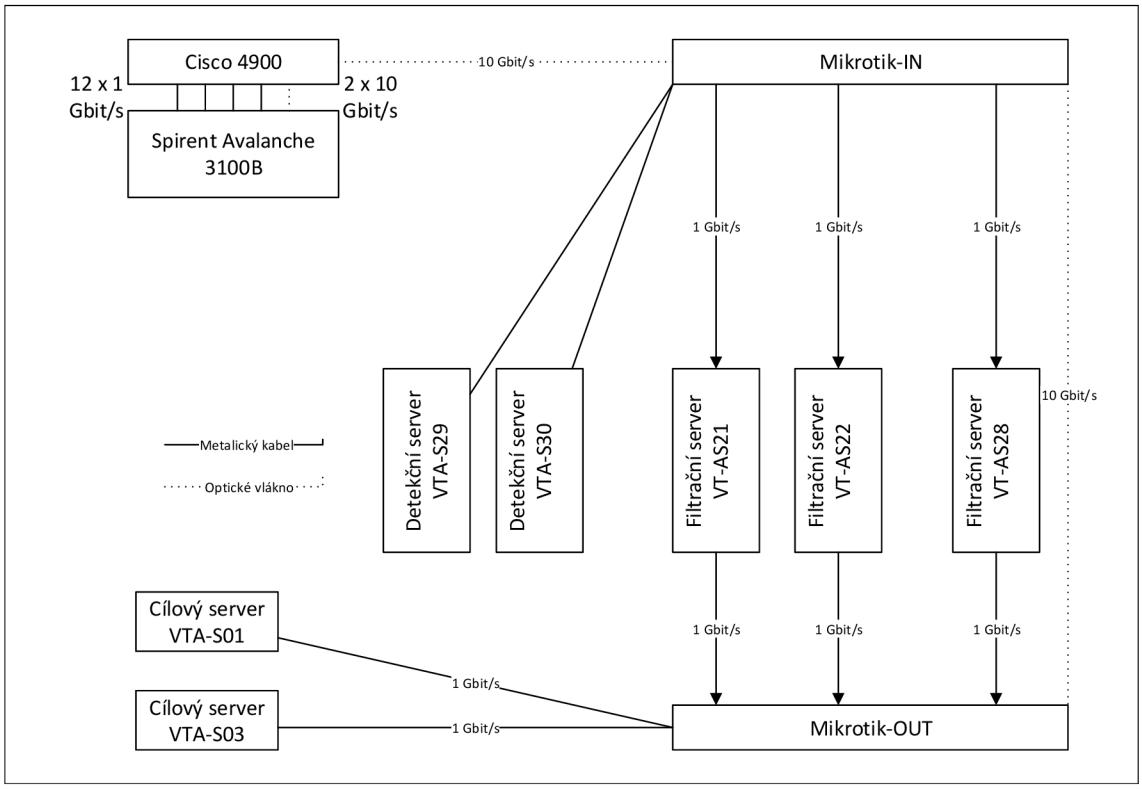
Experimentální pracoviště VUT v Brně slouží k simulaci DDoS útoků pro výzkum a vývoj prostředků k jejich detekci a filtraci. Díky pracovišti je možné i zkoumání dopadů těchto útoků na servery a síťové prvky.

Pracoviště se skládá z několika částí, do kterých jsou jednotlivá zařízení zařazena dle svého účelu. Na obrázku 2.4 je jeho blokové schéma. O vytváření síťového provozu se stará generátor provozu Spirent Avalanche 3100B. Ten může být propojen do switche Cisco 4900 buď pomocí dvanácti metalických spojů, nebo přes dva optické s rychlostí 10 Gbit/s.

Směrování provozu do jednotlivých komponent pracoviště, jako jsou filtrační a detekční servery, zajišťují routery Mikrotik CRS226-24G-2S+. Slouží jako vstupní zařízení pro směrování provozu do filtrační části, současně i jako výstup provozu z tohoto oddílu testovacího pracoviště.

Úlohu filtračních stanic plní osm serverů, přičemž každý je vybaven operačním systémem Debian. Stroje disponují dvěma procesory Intel Xenon E5420. Operační paměť má kapacitu 4 GB. Všechny tyto servery mají několik síťových rozhraní, pro vstup a výstup filtrovaného provozu. Také je zde vždy jedno servisní pro vzdálené připojení a ovládání daného stroje.

Jako detekční servery, které mohou být v rámci testu použity k označení a odklonění provozu generovaného DDoS útokem, slouží dva servery. Vybaveny jsou procesory Intel Xenon E5420. Operačním systémem je zde Debian.
Dva servery plní v testovacích scénářích úlohu cílových serverů. Běží na nich například služby DNS, redakční systém pro weby WordPress, opensource webový server



Obr. 2.4: Schéma experimentálního pracoviště pro simulaci DoS.

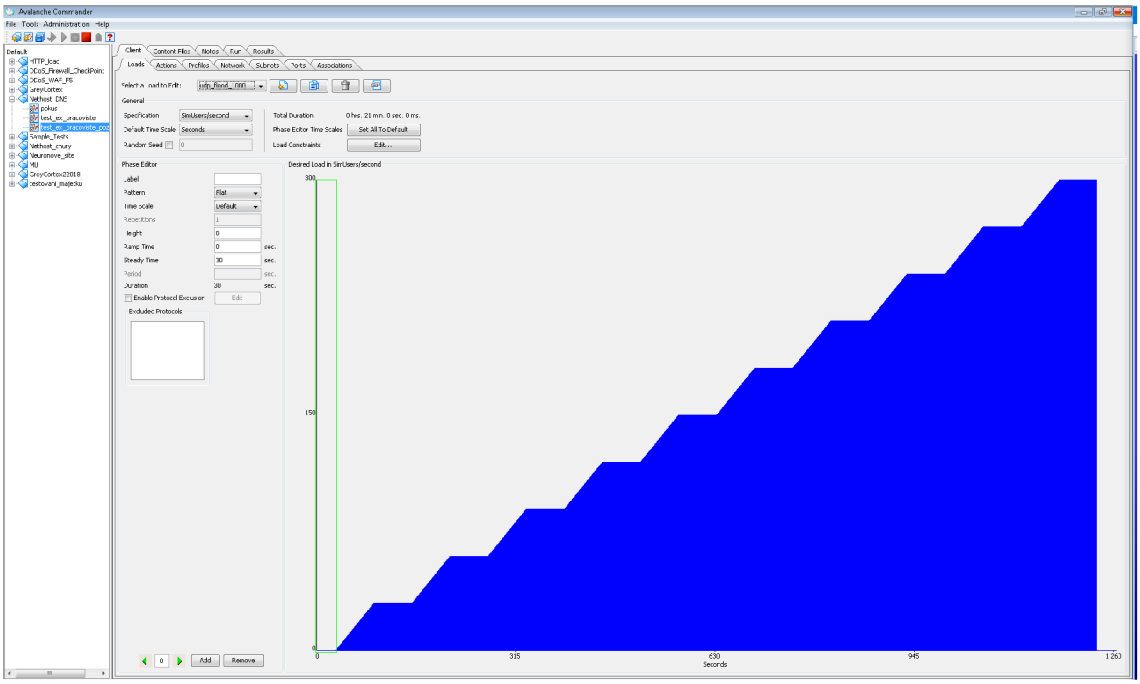
Apache2. Jako operační systém je zde opět použit Debian.

Spirent Avalanche 3100B

Hardwarový generátor síťového provozu pro testování webových aplikací a sítí jednak vzhledem k odolnosti proti DoS útokům, jednak k testování spolehlivosti serverů v reálném provozu bez nutnosti nasazení do ostrého provozu.

Zařízení je schopné generovat více než 1 000 000 HTTP GET požadavků za sekundu [14]. V jeden okamžik je možné udržovat až 30 000 000 [14] sestavených spojení. Z oblasti bezpečnosti síťového provozu je vhodné zmínit i schopnost současného provozu 200 000 [14] tunelovaných spojení pomocí protokolu IPSec. Výrobce zmiňuje schopnost zařízení během testů velmi věrně napodobovat chování reálných uživatelů.

Výstup testovací infrastruktury má čtrnáct portů. Dva z toho jsou optické s rychlostí 10 Gbit/s. Zbylých dvanáct je určeno pro metalickou kabeláž, přičemž maximální rychlosti, které lze dosáhnout na každém z nich, je 1 Gbit/s. Využití optických a metalických portů není možné kombinovat.
K ovládání testeru se používá aplikace Avalanche Commander. Její uživatelské rozhraní je možné vidět na obrázku 2.5. Lze v něm komplexně nastavit mnoho parame-



Obr. 2.5: Aplikace pro ovládání testeru Avalanche 3100B.

trů, pro účely bakalářské práce jsou nejdůležitější zejména profily a akce. Profily v záložce Loads definují průběh simulace v čase, jako například mohutnost generovaného provozu. Záložka Actions definuje dílčí kroky, ze kterých se sestává generovaný provoz. Kupříkladu se může jednat o HTTP GET požadavky.

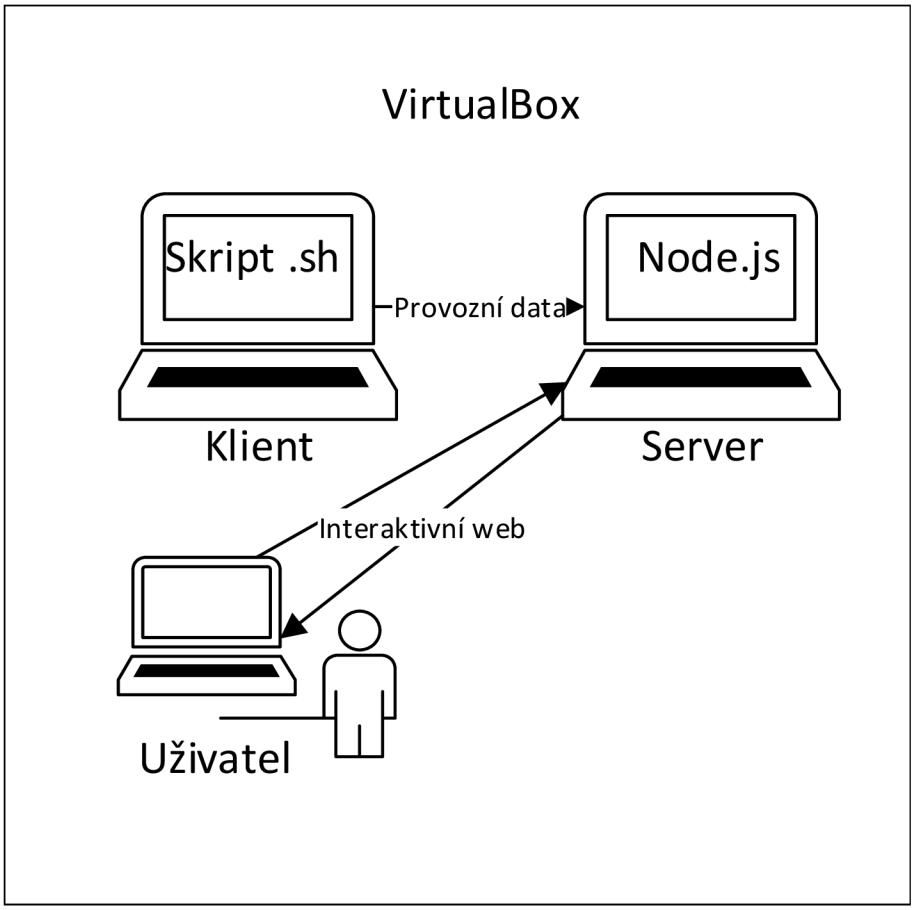
V rámci této bakalářské práce je zátěžový tester použit pro simulaci DDoS útoků na experimentálním pracovišti, které budou zobrazovány v rámci vytvořeného grafického rozhraní. V souladu se zadáním bude využito jeho schopnosti generovat zejména provoz útoků typu SYN Flood, UDP Flood, ICMP Flood, DNS Flood a HTTP Flood. V rámci těchto útoků bude pomocí vytvořeného grafického uživatelského rozhraní zobrazován průběh daných útoků v závislosti na měnících se metrikách těchto útoků.

2.3 Implementace na experimentální pracoviště VUT

Dle zadání má být vytvořené grafické uživatelské rozhraní ve formě interaktivní webové stránky implementované na experimentální pracoviště VUT. V první fázi práce na projektu tak vznikla virtuální infrastruktura, nad kterou probíhal vývoj a základní testovaní. V druhé fázi pak došlo k implementaci všech potřebných komponent na školní experimentální pracoviště a jejich testování.

Virtuální infrastruktura pro vývoj a testování

Pro vývoj a testování webové stránky, která slouží jako grafické rozhraní systému filtrace DoS, je použito několika virtuálních strojů, jak je vidět na obr. 2.6. Ty jsou provozovány pomocí programu VirtualBox 5.2.20. Jedná se o multiplatformní virtualizační nástroj, vyvíjený společností Oracle pod licencí GNU GPLv2. Z toho plyne, že se jedná o svobodný software. Umožňuje provozování mnoha typů operačních systémů. Například Windows, MAC OS X a pro tento projekt zejména různé linuxové distribuce. Dále jde tvořit i virtuální sítě, této funkcionality je opět v projektu hojně využíváno. V rámci práce na projektu jsou vytvořeny dva virtuální stroje. Oba běží



Obr. 2.6: Virtuální infrastruktura pro vývoj a testování grafického rozhraní.

na 64-bit OS Debian ve verzi 9. Každý stroj má 2 GB RAM a 20 GB úložiště. Na jednom z nich jsou sbírána data o využití procesoru, paměti RAM a o množství informace, která projde přes síťová rozhraní. Tato stanice ve virtuálním prostředí simuluje filtrační server, ve schématu je nazvána jako klient. Data z klienta jsou posílána pomocí protokolu SSH na druhou stanici. Ta je ve schématu nazvána jako server. Sbírá data z klienta a provozuje serverovou aplikaci, založenou na technologii Node.js. Serverová aplikace pak hostuje interaktivní webovou stránku.

Nasazení projektu na experimentální pracoviště VUT

Nasazení projektu do reálného provozu mělo několik fází. Veškeré konfigurace na experimentálním pracovišti 2.4 byly prováděny vzdáleně díky připojení přes VPN a s použitím SSH klienta.

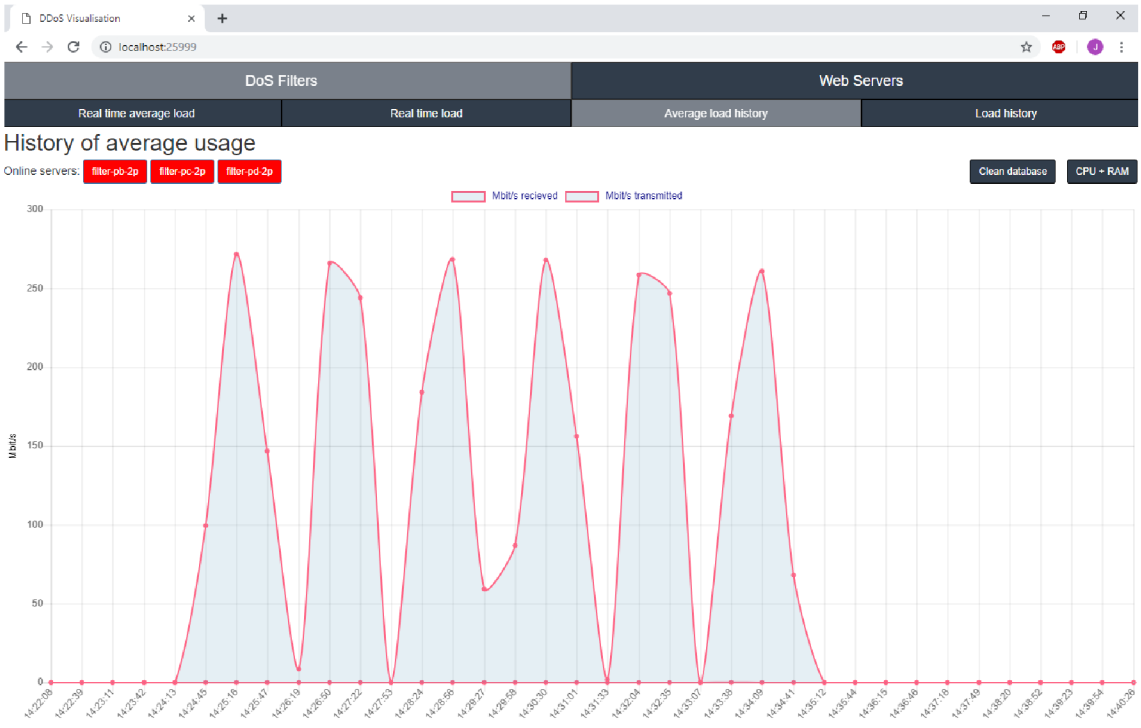
Nejprve byly po konzultaci s vedoucím práce vybrány stanice, na které měl být projekt nainstalován. Následně byl na těchto strojích vytvořen uživatelský účet, pod nímž je možné projekt provozovat. Jako zástupci filtračních serverů jsou zvoleny stanice VT-AS26, VT-AS27, VT-AS28. Zástupce cílových serverů tvoří VT-AS01 a VT-AS03. Roli webového serveru plní pak Detect-B. Dalším krokem bylo vytvoření obousměrného SSH spojení bez nutnosti zadávání přihlašovacích údajů pro vytvořený účet mezi všemi stanicemi a strojem, který plní roli webového serveru. Na všechny sledované stroje byly nahrány skripty pro logování dat. Každý skript musel být upraven na míru danému stroji. Na Detect-B byla nainstalována technologie Node.js a nahrána složka s řešením projektu, která obsahuje zdrojový kód webového serveru a potřebné soubory interaktivní webové stránky. Poté už k uvedení projektu do provozu stačilo spustit logovací skripty a zapnout Node.js webový server.

2.4 Testování aplikace

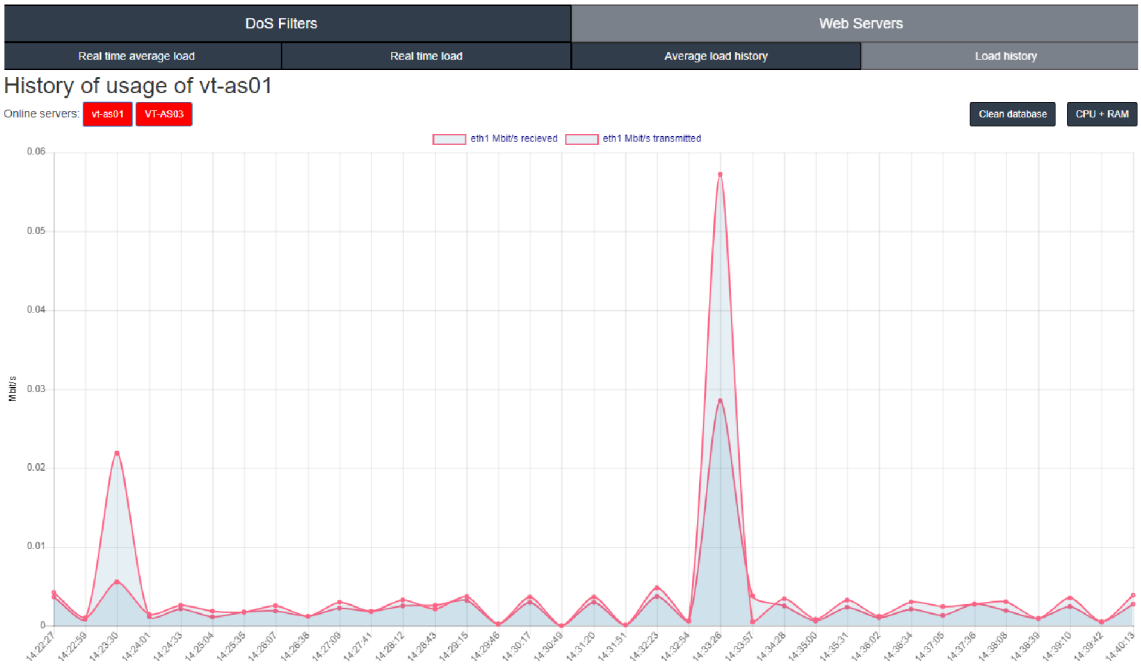
Podle zadání má být pomocí vzniklého řešení vizualizováno několik různých útoků na experimentálním pracovišti VUT. V rámci testování projektu byly realizovány útoky typu SYN Flood, ICMP Flood, DNS Flood a HTTP Flood. Veškeré útoky byly cíleny na stanice, které byly jmenovány v předchozí podkapitole a sběr dat probíhal právě z těchto stanic.

Útok typu SYN Flood měl střídavý průběh. Tester Spirent Avalanche 3100B střídal intervaly generování maximálního provozu s intervaly bez síťového provozu. Celková zátěž všech filtračních serverů činila přes 270 Mbit/s. Během tohoto testu byl pomocí iptables blokován odchozí provoz ze všech filtračních serverů a tak, jak je vidět i na obrázku 2.8, je na cílových serverech pracoviště minimální síťový provoz, jenž je tvořen pouze běžným provozem v síti. Příčinou ne zcela přesného vykreslení grafu střídání zátěže ze strany testeru na obrázku 2.7 je zvolená metodika měření. Data jsou sbírána v reálném čase z několika stanic, a proto bylo z důvodů optimalizace při vývoji sáhnuto ke sběru po půlminutových intervalech a následnému průměrování dat z těchto intervalů za účelem získání rychlosti přenosu za sekundu. Celý útok trval jedenáct minut.

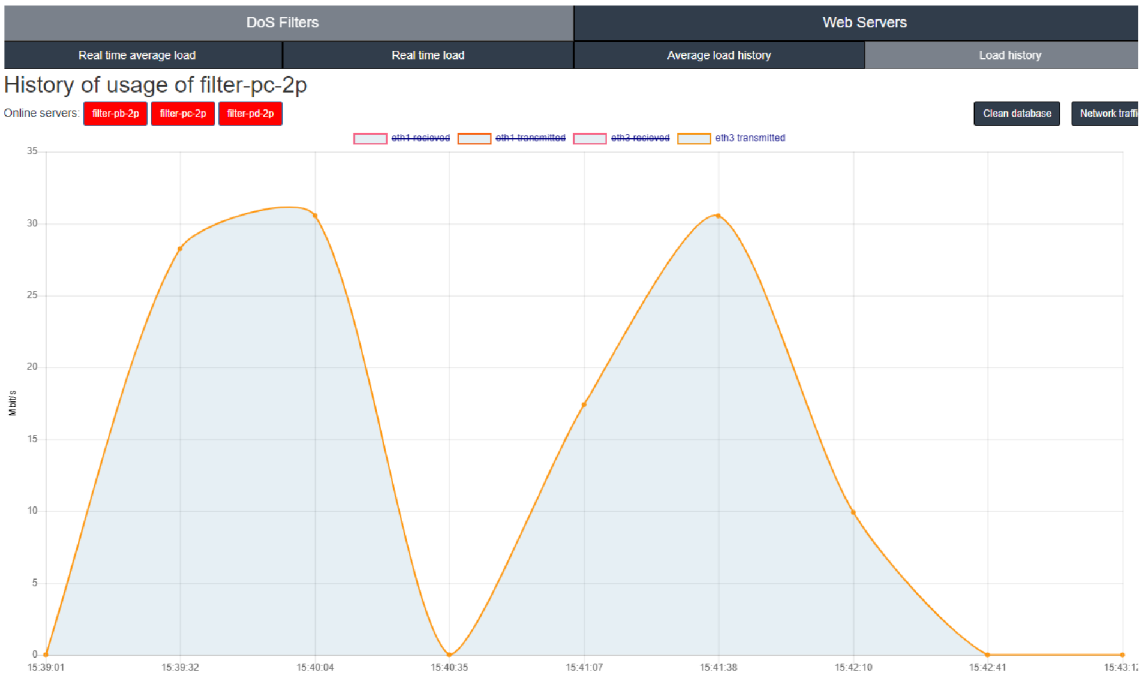
Dalším testovaným útokem byl čtyřminutový ICMP Flood. Spirent Avalanche 3100B opět generoval provoz střídavě. Minutové intervaly, během nichž každý z filtračních serverů zaplavovala příchozí data rychlostí 32 Mbit/s, střídaly čtyřiceti



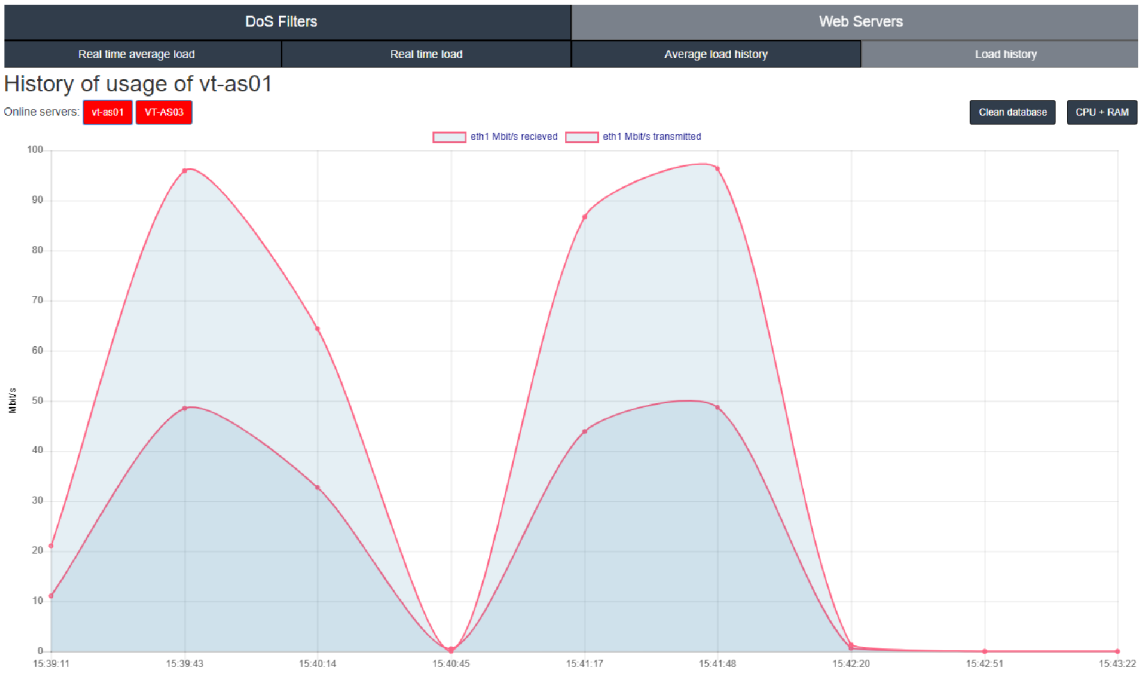
Obr. 2.7: Součet vytížení všech síťových rozhraní filtračních serverů během SYN Flood.



Obr. 2.8: Provoz na síťovém rozhraní cílového serveru VT-AS01 během SYN Flood.

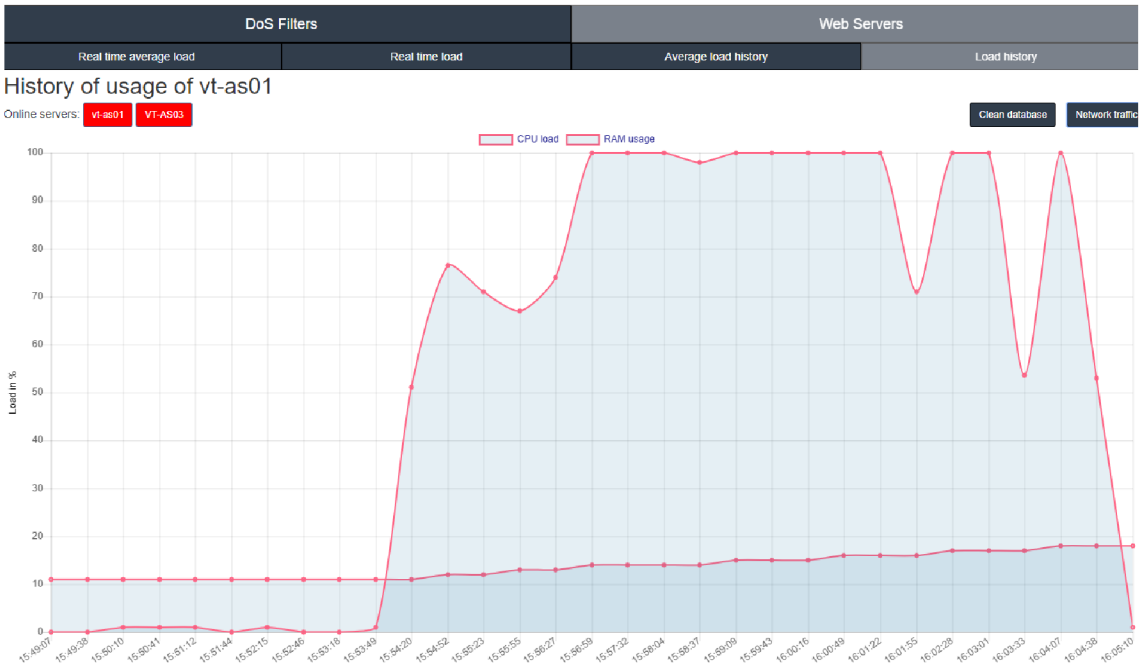


Obr. 2.9: Využití síťového rozhraní serveru filter-pc-2p při útoku ICMP Flood.



Obr. 2.10: Provoz na síťovém rozhraní cílového serveru VT-AS01 během ICMP Flood.

pěti sekundové intervaly klidu. Tentokráte byly filtrační servery nastaveny tak, aby provoz přeposílaly, jak je vidět na obr. 2.9. Cílem takového provozu byl pak server VT-AS01 na obrázku 2.10, který přijímal data rychlostí přes 90 Mbit/s. Nepřesnosti

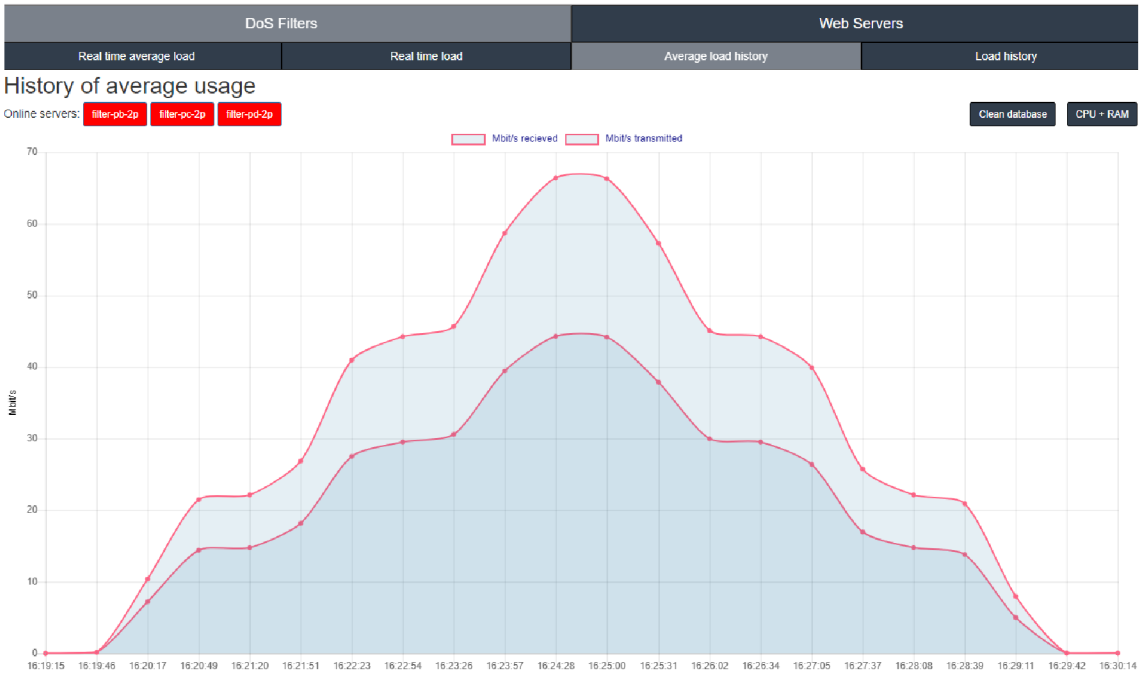


Obr. 2.11: Využití výpočetní kapacity cílového serveru VT-AS01 během ICMP Flood.

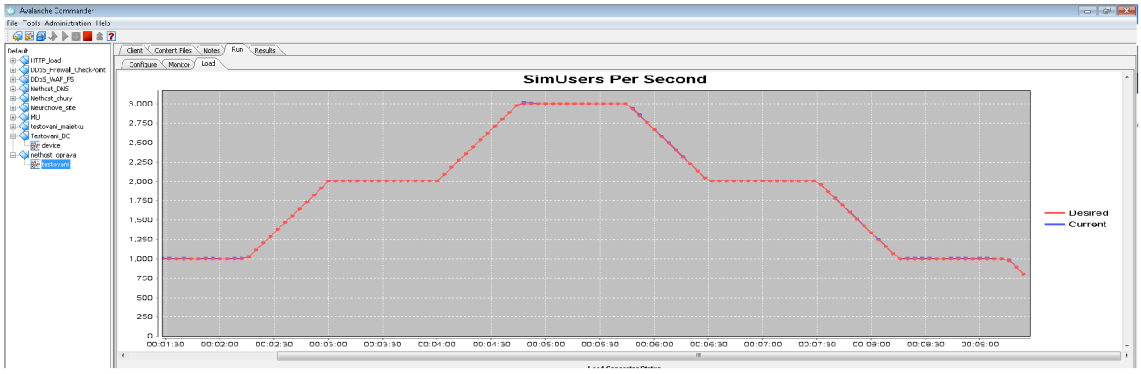
ve vykreslení grafů jsou způsobeny metodikou sběru dat.
Na útok HTTP Flood reagovaly oba cílové servery. Mohutnost útoku byla okolo 80 Mbit/s. Jak je vidět na obrázku 2.11, jeden z nich, VT-AS01, byl útokem dokonce zahlcen. Během testu byl chvílemi jeho stav indikován jako offline. Docházelo k nárůstu využití RAM postupně z jedenácti na devatenáct procent. Markantnější změny nastaly ve vytížení procesoru. To dosahovalo hodnot okolo sta procent. Když uvážíme, že naměřené hodnoty reprezentují vždy průměr z třiceti sekund provozu, vyplývá z měření, že takové hodnoty vytížení procesoru musely být platné po celý měřený interval. Server tedy fungoval na hranici svých možností. V tomto případě filtrační servery přeposílaly veškerý provoz dál na cílové servery.

Posledním zde zmíněným útokem je DNS Flood. Jeho průběh byl postupně schodovitě stoupající a odpovídal nastavení zátěžového testeru. Po dosažení vrcholu v necelých 70 Mbit/s nastal návrat zpět k nulovým hodnotám. Test trval po dobu deseti minut. Zajímavé je porovnat průběh útoku na obrázku 2.13 s hodnotami 2.12, které byly naměřeny řešením, realizovaným v rámci této práce. Jsou patrné limity projektu, zejména v oblasti sběru dat v reálném čase. Avšak na druhou stranu je vidět, že grafy mají dostatečnou vypovídací hodnotu pro zjištění stavu experimentálního pracoviště.

Grafické rozhraní, které vzniklo v rámci této bakalářské práce, bylo vyvíjeno a testováno pro webový prohlížeč Google Chrome ve verzi 73. Využití paměti



Obr. 2.12: Využití síťových karet filtračních serverů při DNS Flood.



Obr. 2.13: Grafické rozhraní řízení testeru Spirent Avalanche 3100B během DNS Flood.

JavaScriptem, díky kterému je webová stránka rozhraní interaktivní, dosahuje průměrně hodnot okolo 5 MB. Co se týče rychlosti odezvy vytvořeného serveru, zde velmi záleží na kvalitě připojení klienta. Tj. některé méně náročné požadavky jsou zpracovány serverem o něco rychleji - například požadavek na data o využití jednoho serveru bude za předpokladu stejného připojení k Internetu zpracován nepatrně rychleji, než požadavek na data o všech sledovaných serverech. Řádově jsou však odpovědi serveru doručeny v řádu desítek, maximálně nižších stovek milisekund. Řešení je výkonné i v případě vykreslení grafu s velkým množstvím záznamů.

3 Závěr

Cílem bakalářské práce bylo navrhnout a vytvořit grafické uživatelské rozhraní pro systém filtrace kybernetických útoků na odepření přístupu ke službě DoS. Toto rozhraní má být implementováno na experimentální pracoviště VUT a jeho výstup by měl umožnit analýzu průběhu DoS útoku a jeho dopadů na celý sledovaný systém. Rozhraní vytvořené v rámci této práce by mělo mít formu interaktivní webové stránky. Dále bylo úkolem seznámení s experimentálním pracovištěm VUT. V neposlední řadě pak i studium teorie související s tímto projektem.

Definované cíle byly splněny. V rámci bakalářské práce byl navržen a implementován komplexní systém, jehož výstupem je grafické uživatelské rozhraní, vhodné k analýze průběhu útoků DoS na experimentálním pracovišti VUT. Řešení se skládá ze skriptů pro sběr dat, speciálně vytvořeného webového serveru a interaktivní webové stránky pro grafickou interpretaci naměřených hodnot přímo v prohlížeči klienta.

Během konzultací k bakalářské práci došlo k seznámení s infrastrukturou školního experimentálního pracoviště. Zajímavou zkušeností byla možnost poznat i velmi komplexní ovládací rozhraní testeru Spirent Avalanche 3100B. Poté, co proběhlo proškolení, bylo možné díky zabezpečenému vzdálenému přístupu přes VPN pracovat s vybavením pracoviště a implementovat projekt přímo na jeho servery.

Poslední fází práce na projektu bylo testování. V rámci něj byl vyzkoušen provoz celého systému v případě reálné zátěže. Z dosažených výsledků bylo zjištěno, že navržené a implementované řešení je funkční, naměřené hodnoty odpovídají reálnému provozu. Samozřejmě přesnost nebyla úplná, to proto, že sběr dat v reálném čase z více samostatných stanic je poměrně složitým problémem. Odchylky od reality vycházely z metodiky měření krátkých intervalů, z nichž byla počítána průměrná rychlost přenosu.

Silnou stránkou řešení je vyvinutý Node.js webový server. Ten se v praxi ukázal být velmi výkonný a rychlý. Samotné grafické uživatelské rozhraní v podobě interaktivní webové stránky je přehledné a intuitivní. Z hlediska výkonu řešení je třeba vyzdvihnout i fakt, že veškeré grafy a grafické prvky jsou tvořeny dynamicky pomocí JavaScriptu v prohlížeči klienta, tím pádem dochází k úspoře prostředků webového serveru. Ten má za úkol pouze zpracování dat a pro každé načtení výsledné webové stránky jednorázové odeslání souborů stránky klientovi.

Literatura

[1] BHUYAN, Monowar H.; BHATTACHARYYA, D. K.; KALITA, Jugal K. An empirical evaluation of information metrics for low-rate and high-rate DDoS attack detection. Pattern Recognition Letters, 2015.

[2] Palo Alto Networks: *What is a denial of service attack* [online]. [cit. 27. 10. 2018]. Dostupné z URL: <https://goo.gl/e3xf5z>.

[3] *DDOS ATTACKS. Incapsula* [online]. [cit. 2018-10-30]. Dostupné z: Dostupné z URL: <https://www.incapsula.com/ddos/ddos-attacks.html/>.

[4] WESLEY, M. Eddy. Defenses Against TCP SYN Flooding Attacks. Cisco [online]. 2006, December 2006 [cit. 2018-11-02].
Dostupné z: <https://goo.gl/r3G5hH>.

[5] *UDP Flood. DDOS-GUARD* [online]. [cit. 2018-11-02]. Dostupné z:
<https://ddos-guard.net/en/terminology/udp-flood/>.

[6] *Anatomy of an HTTP Transaction. Nodejs.org* [online]. [cit. 2019-04-25]. Dostupné z:
<https://nodejs.org/uk/docs/guides/anatomy-of-an-http-transaction/>.

[7] *Node.js modules. Nodejs.org* [online]. [cit. 2019-04-25]. Dostupné z:
<https://nodejs.org/api/modules.html#modules_modules/>.

[8] ROUSE, Margaret. *ICMP (Internet Control Message Protocol)* [online]. [cit. 2019-04-05]. Dostupné z:
<https://searchnetworking.techtarget.com/definition/ICMP/>.

[9] CHRZANOWSKA, Natalia. *12 Top Applications Written in Node.js - Examples from Big Companies. (Netguru)* [online]. [cit. 2019-04-07]. Dostupné z:
<https://www.netguru.com/blog/top-companies-used-nodejs-production/>.

[10] D-WARD [online]. [cit. 2019-05-13]. Dostupné z:
<https://lasr.cs.ucla.edu/dward//>.

[11] *DNS flood DDoS attack. (Cloudflare)* [online]. [cit. 2019-04-05]. Dostupné z:
<https://www.cloudflare.com/learning/ddos/dns-flood-ddos-attack/
/>.

[12] *A brief history of Node.js. (Node.js)* [online]. [cit. 2019-04-06]. Dostupné z:
<https://nodejs.dev/a-brief-history-of-nodejs//>.

[13] Kaspersky DDoS protection data sheet [online]. [cit. 2019-05-13]. Dostupné z: <https://media.kaspersky.com/kaspersky-ddos-protection-data-sheet.pdf//>.

[14] SPIRENT AVALANCHE 3100B. Infopoint Security [online]. 2011 [cit. 2018-12-06]. Dostupné z: <https://www.infopoint-security.de/medien/spirent_avalanche_3100_datasheet.pdf>

[15] VirtualBox [online]. [cit. 2018-12-06]. Dostupné z: <https://www.virtualbox.org/>.

[16] About Wireshark [online]. [cit. 2018-11-09]. Dostupné z: <https://www.wireshark.org/>.

[17] ROUSE, Margaret. [online]. [cit. 2018-11-09]. Dostupné z: <https://www.theserverside.com/definition/JavaScript>.

[18] Canvas tutorial [online]. [cit. 2018-11-09]. Dostupné z: <https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial>

[19] DE ROSA, Aurelio. Creating Beautiful Charts with Chart.js [online]. January 07, 2015 [cit. 2018-11-09]. Dostupné z: <https://www.sitepoint.com/creating-beautiful-charts-chart-js/>.

[20] JQuery [online]. [cit. 2018-12-07]. Dostupné z: <https://jquery.com/>

[21] Using Google Charts [online]. únor 23, 2017 [cit. 2018-11-09]. Dostupné z: <https://developers.google.com/chart/interactive/docs/>

[22] Volební weby byly nedostupné kvůli DDoS útoku [online]. 2018 [cit. 2018-12-07]. Dostupné z:<https://www.czso.cz/csu/czso/volebni-weby-byly-nedostupne-kvuli-ddos-utoku>

[23] BENNETT, Michael. Zero-Day DDoS Attacks? What?. In: Medium.com [online]. Nov 1, 2016 [cit. 2019-05-12]. Dostupné z: <https://medium.com/@bennettaur/zero-day-ddos-attacks-what-93f57e0e2d6>

[24] Flot [online]. [cit. 2018-11-09]. Dostupné z: <https://github.com/flot/flot/blob/master/API.md#introduction>

[25] BROWN, Bob. Carnegie Mellon researchers visualize way to fend off DDoS attacks. NETWORKWORLD [online]. NOV 8, 2016 [cit. 2018-11-02]. Dostupné z: <https://goo.gl/2vbX8z>.

[26] BANERJEE, Souvik. Introduction of Website Templates for Beginners [online]. March 5, 2013 [cit. 2018-11-09]. Dostupné z: <https://www.rswebsols.com/tutorials/web-design/website-templates-introduction-for-beginners>

[27] RAGAN, Steve. 8 top cyber attack maps and how to use them. CSO [online]. AUG 21, 2017 [cit. 2018-11-02]. Dostupné z: <https://goo.gl/2GKPYm>.

[28] HTTP Flood. Imperva [online]. [cit. 2019-05-13]. Dostupné z: <https://www.imperva.com/learn/application-security/http-flood/>.

Seznam symbolů, veličin a zkratek

IoT	Internet of Things
DoS	Odepření služby z anglického Denial of Service
DDoS	Distributed Denial of Service
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secured
ICMP	Internet Control Message Protocol
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
DNS	Domain Name System
IDS	Intrusion Detection System
IPS	Intrusion Prevention Systems
DOM	Document Object Model
SSH	Secure Shell

Seznam příloh

A Uživatelská příručka

A.1 Spuštění

Přihlášení probíhá vždy pomocí přihlašovacích údajů ze souboru **udaje.txt** elektronické přílohy.

1. Připojení pomocí SSH klienta na IP adresu 12.12.13.29 a přihlášení pomocí přihlašovacích údajů.
2. Přepnutí do adresáře */projekt/myapp*.
3. Spuštění Node.js serveru příkazem:
   `jb@detect-b: /projekt/myapp$ node SERVERapp.js`
4. Připojení pomocí SSH klienta na IP adresy servisních rozhraní stanic VT-AS26, VT-AS27, VT-AS28, VT-AS01, VT-AS03 a přihlášení pomocí přihlašovacích údajů.
5. Přepnutí do adresáře */projekt* na každé stanici.
6. Spuštění sběru dat na každé stanici příkazem:
   `jb@filter-pb-2p: /projekt$ sh sber.sh`
7. Ukončením jednotlivých skriptů pro sběr dat a ukončením Node.js serveru dojde k vypnutí projektu grafického rozhraní.

A.2 Obsluha

K webové stránce grafického uživatelského rozhraní lze přistoupit zadáním adresy **http://12.12.13.29:25999/**, a to včetně čísla portu, do webového prohlížeče. Kliknutím na záložky DoS Filters a Web Servers je možné přepínat mezi vizualizací využití filtračních a cílových serverů. Tlačítka o řádek níže pak nastavují, zda mají být zobrazena agregovaná data, případně data z jednotlivých stanic. Barevné ikonky s názvy sledovaných serverů slouží k indikaci, zda jsou dané servery online. V případě, že se uživatel nachází v sekci Real time load a Load history, fungují současně jako voliče serveru ke sledování a v sekci Load history i jako tlačítko k vykreslení požadovaného grafu. Pokud se uživatelské rozhraní nachází ve stavu, kdy vyžaduje od uživatele časový údaj, jsou zobrazena textová pole ve spodní části stránky. V rámci těchto textových polí je třeba striktně dodržet formát zadávání časového údaje. Pokud ho uživatel nedodrží, případně zadá časový údaj, který se v databázi systému nenachází, vykreslí se prázdný graf. V případě grafů v reálném čase je pro offline servery použita poslední známá hodnota. Tlačítko s nápisem CPU+RAM či Network traffic má za úkol přepnutí zobrazení.

B Obsah přiloženého CD

- `/` ..... kořenový adresář přiloženého CD
  - `udaje.txt` ..... přihlašovací údaje
  - `BakalarskaPrace.pdf` ..... elektronická verze práce ve formátu PDF
  - `sber.sh` ..... skript pro sběr dat na sledovaných stanicích
  - `myapp` ..... kořenový adresář vytvořeného webového serveru
    - `app.js` ..... hlavní JavaScript soubor klientské části
    - `bootstrap.min.css` ..... knihovna bootstrap
    - `Chart.bundle.min.js` ..... knihovna chart.js
    - `index.html` ..... HTML dokument vizualizační stránky
    - `jquery-3.3.1.min` ..... knihovna jQuery
    - `Moment.js` ..... knihovna moment.js
    - `SERVERapp.js` ..... zdrojový kód Node.js serveru
    - `style.css` ..... soubor CSS stylů webové stránky
    - `zaznamy` ..... složka pro nasbíraná měřená data