VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2020 Michael Jurek

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

DETEKCE DOS ÚTOKŮ NA PROTOKOL HTTP/2
DETECTION OF HTTP/2 DOS ATTACKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
Michael Jurek

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Marek Sikora

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Michael Jurek ***ID:*** 182503
Ročník: 3 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Detekce DoS útoků na protokol HTTP/2

POKYNY PRO VYPRACOVÁNÍ:

Nová verze protokolu HTTP přináší mnohá vylepšení, zvýšení efektivity datových přenosů a zvýšení úrovně zabezpečení. Nicméně, nově použité mechanismy otevírají také nové možnosti útočníkům, kteří hledají další slabiny protokolu.
Student v rámci této bakalářské práce analyzuje a popíše aktuální známé hrozby a útoky na protokol HTTP/2 a následně vytvoří testovací prostředí. Jeho hlavním úkolem je vytvořit generátor a modely několika typů útoků. Dalším úkolem je testování jejich účinnosti vůči běžně používaným webovým serverům. Posledním hlavním úkolem je návrh a testování metodiky pro detekci těchto útoků.

DOPORUČENÁ LITERATURA:

[1] TRIPATHI, Nikhil a Neminath HUBBALLI. Slow rate denial of service attacks against HTTP/2 and detection. Computers & Security [online]. 2018, 2018(72), 255-272 [cit. 2019-04-18]. DOI: 10.1016/j.cose.2017.09.009. ISSN 01674048. Dostupné z: https://bit.ly/2lKGrjM

[2] ADI, Erwin, Zubair BAIG a Philip HINGSTON. Stealthy Denial of Service (DoS) attack modelling and detection for HTTP/2 services. Journal of Network and Computer Applications [online]. 2017, 91, 1-13 [cit. 2019-09-11]. DOI: 10.1016/j.jnca.2017.04.015. ISSN 10848045. Dostupné z: https://bit.ly/2kgDI1i

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

Vedoucí práce: Ing. Marek Sikora

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

ABSTRAKT

Bakalářská práce se zaměřuje na problematiku generování a detekce síťových útoků Slow DoS aplikačního protokolu HTTP/2. Má za cíl shrnout základní fungování počítačových sítí a vývoj webových protokolů a přiblížit principy útoků. V rámci této práce je navrhnut a implementován nástroj pro generování Slow DoS útoků, kerý je otestován na webovém serveru Apache 2.4.17. Součástí práce je návrh a implementace systému detekce Slow DoS útoků.

KLÍČOVÁ SLOVA

Pomalé DoS útoky, Slow READ, Slow POST, Slow PREFACE, Slow HEADERS, Slow SETTINGS, hyper, h2, hyperframe, detekce, HTTP/2, pyshark, Apache 2.4.17, generátor

ABSTRACT

This bahelor thesis is focused on the issue of generating and detecting network Slow DoS attacks of application protocol HTTP/2. As a goal it has to sum up princples of networking, chronological development of web protocols and principles of these attacks. It provides design, creation and testing of Slow DoS attacks generator and theirs successfull detection. Final product is tested on real web server Apache 2.4.17.

KEYWORDS

Slow DoS attacks, Slow READ, Slow POST, Slow PREFACE, Slow HEADERS, Slow SETTINGS, hyper, h2, hyperframe, detection, HTTP/2, pyshark, Apache 2.4.17, generator

JUREK, Michael. *Detekce DoS útoků na protokol HTTP/2*. Brno, 2020, 91 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Marek Sikora

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Detekce DoS útoků na protokol HTTP/2“ 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 diplomové práce panu Ing. Marku Sikorovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Dále bych chtěl poděkovat své rodině za trpělivost a veškerou podporu při studiu.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Tato bakalářská práce se zabývá problematikou počítačových útoků cílených na odepření služby (DoS – Denial of Service) webového serveru pomocí Slow (Pomalých) DoS útoků a protokolu aplikační vrstvy HTTP/2 (Hypertext Transport Protocol 2. generace). Obecně DoS útoky, kvůli jednoduché realizaci, dnes patří mezi nejčastěji realizované a cílí zejména na webové servery. Přes svoji jednoduchost však mohou mít katastrofální důsledky pro provozovatele webových serverů. Tato oblast kybernetické bezpečnosti se vyvíjí velmi dynamicky. Dnes se stírají rozdíly v přesných definicích DoS a DDoS (Distributed Denial of Service) útoků. Útočníci volí kombinaci více způsobů pro co největší dosah útoků, ať do výše škod, nebo do počtu napadených zařízení. I přes relativně špatnou detekci těchto útoků je důležité se jimi zabývat.

Cílem práce je implementovat a popsat Slow DoS útoky spolu s následky provedení těchto útoků na vybrané webové servery (webservery). V první kapitole je popsán model komunikace spolu s principem zahájení síťové komunikace.

Druhá kapitola popisuje principy protokolu HTTPx a porovnává všechny významné verze. Konkrétně se zabývá protokolem HTTP/2. Tento protokol je klíčový k provedení Slow DoS útoků.

Třetí kapitola se zabývá samotnými DoS a DDoS útoky. Popisuje jejich rozdíly v rámci vrstvy modelu ISO/OSI nebo z pohledu typu útoku.

Čtvrtá kapitola se zabývá popisem a následky provedení jednotlivých Slow DoS útoků. Jedná se konkrétně o Slow READ, Slow POST, Slow PREFACE, Slow HEADERS, Slow SETTINGS. Dále porovnává účinky těchto útoků na jednotlivé webové servery.

Pátá kapitola popisuje testovací prostředí, ve kterém byly tyto útoky testovány. Dále poskytuje zjednodušený návod na sestavení webového serveru ze zdrojových balíčků a metodu detekce dostupnosti webového serveru.

Šestá kapitola uvádí návrh a implementaci generátoru Slow DoS útoků. Popisuje aplikaci, její fungování, vstupní parametry, ale i reprezentaci výsledků a tvorbu grafů.

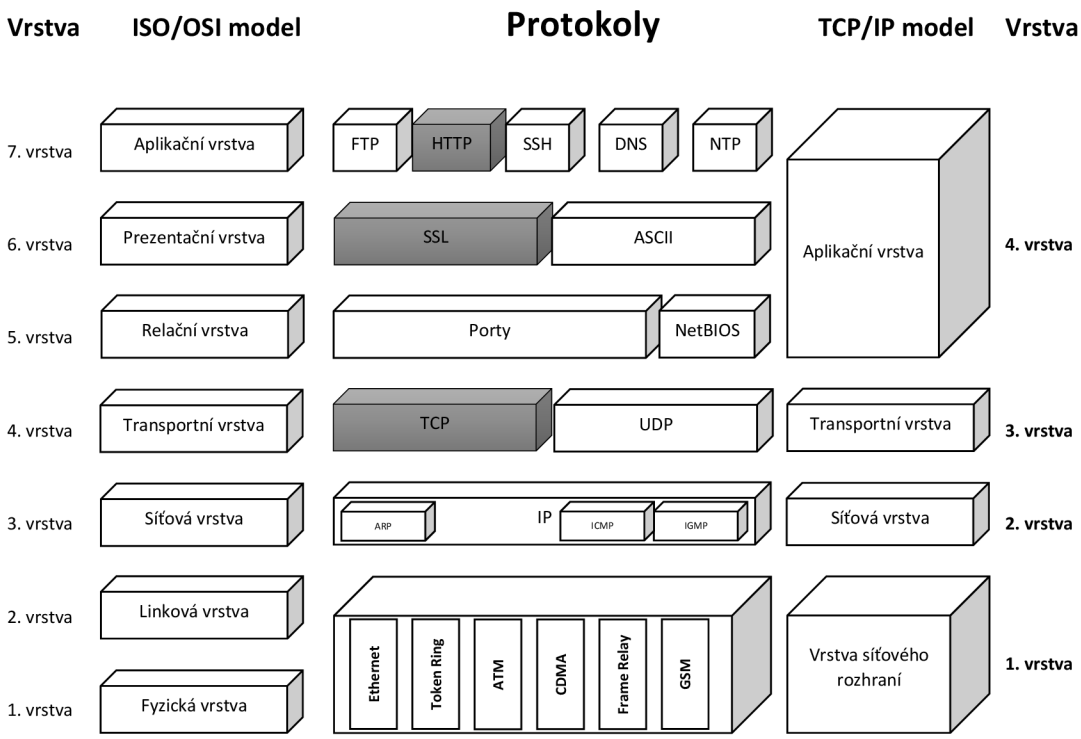
Sedmá kapitola se zabývá detekcí Slow DoS útoků a popisuje obecné metody obrany. Obsahuje návrh a implementaci detektoru Slow DoS útoků. Popisuje vlastnosti útoků, shrnuje funkcionalitu aplikace, udává vstupní parametry aplikace a na závěr interpretuje výsledky detekce.

1 Síťová komunikace

S rozvojem počítačových sítí, nejprve experimentálních, následně univerzitních a privátních, vznikla potřeba standardizace síťového provozu. Základní myšlenky byly:

- Každá vrstva by měla zajišťovat určité funkce s možností další standardizace.
- Jednotlivá vrstva by měla umožňovat dostatečnou abstrakci.
- Přechod mezi jendotlivými vrstvami by měl být volen tak, aby tok dat byl co nejmenší.
- Vzniklo 7 vrstev modelu ISO/OSI.[1]

Jedná se o vrstevnatý model, kdy každá vrstva komunikuje se sousední. Každá vrstva přidává záhlaví (headers) dané vrstvy, případně kontrolní součet pro zajištění integrity dat (obr. 1.1). Dnes se ovšem v praxi více používá model TCP/IP, který se liší od ISO/OSI tím, že na úrovni aplikační vrstvy shlukuje relační, prezentační a aplikační vrstvy modelu ISO/OSI a fyzickou a linkovou vrstvu ve vrstvu síťového rozhraní.



Obr. 1.1: Síťový model ISO/OSI a TCP/IP

[1]Standard ISO 7498, CCIT doporučení X.200

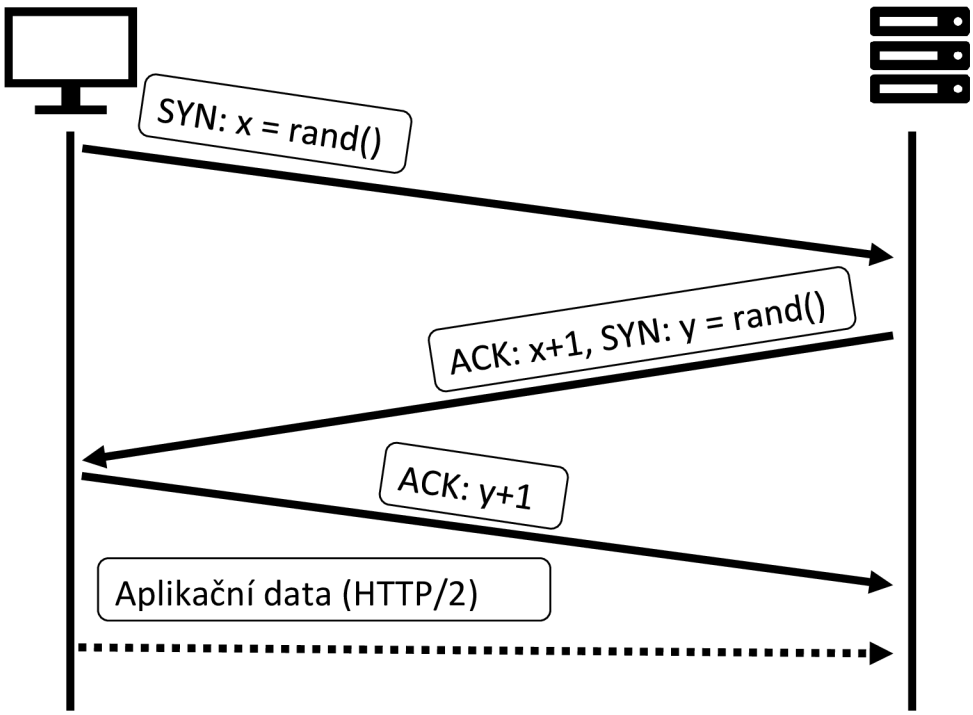
1.1 Zahájení komunikace – 3-way handshake

Uvažujme zahájení komunikace na transportní vrstvě. K tomu je využit protokol TCP (Transmission Control Protocol), který garantuje spolehlivé doručení zpráv ve správném pořadí. Spojuje sousední síťovou a aplikační vrstvu modelu TCP/IP.

Před samotným navázáním spojení protokolem na aplikační vrstvě (HTTP – Hypertext Transport Protocol), musí dojít k navázání na všech nižších vrstvách. Na transportní vrstvě dochází k navázání spojení metodou 3-way handshake, viz obr. 1.2.

1.1.1 Princip navázání spojení

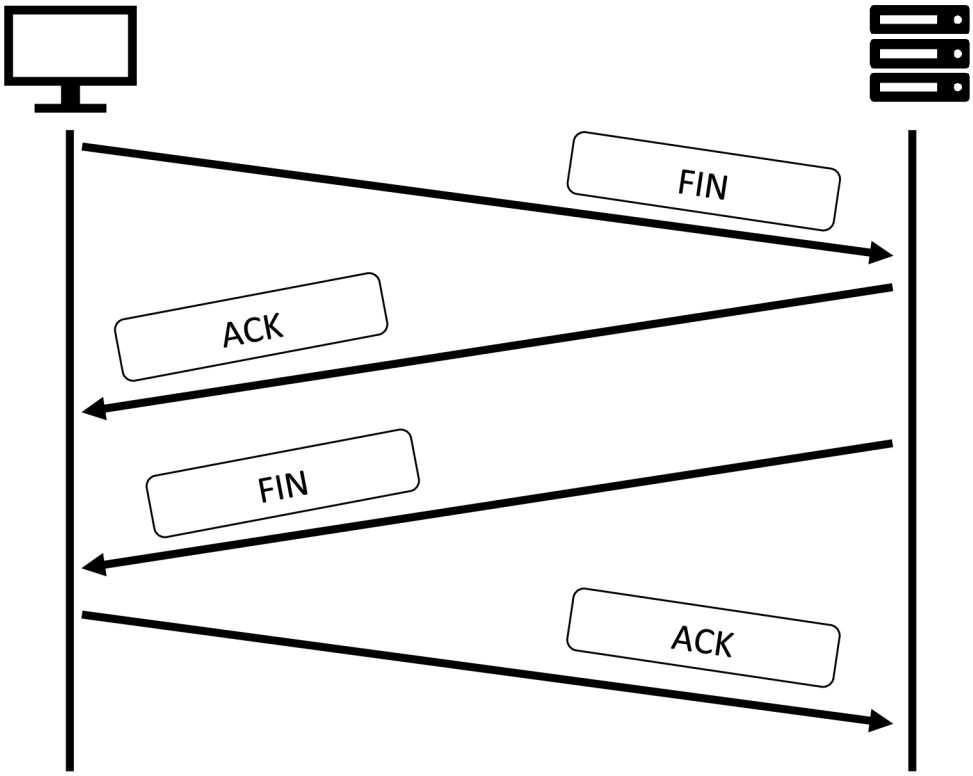
Každá z komunikujících stran si zvolí náhodně pořadové číslo. Klient odešle datagram s příznakem `SYN` a náhodně zvolené číslo x. Server odešle klientovi datagram s příznakem `SYN`, `ACK`, potvrzovací číslo $x+1$ a náhodně vygenerované y. Klient odešle datagram s příznakem `ACK`, pořadové číslo $x+1$ a číslo odpovědi $y+1$. Tato čísla se používají pro další komunikaci a určují následné pořadí zpracování paketů. [1]



Obr. 1.2: TCP 3-way handshake

Ukončit TCP spojení lze pomocí 4-way handshaku standardním, nebo vynuceným způsobem. Standardní způsob spočívá v tom, že každá strana pošle pár pří-

znaků `FIN` a `ACK`, které znamenají vynucené ukončení spojení, inicializující stranou, a následné potvrzení protistranou. (obr. 1.3). [2]



Obr. 1.3: Standardní ukončení TCP spojení

Druhý způsob ukončení TCP spojení je ve chvíli, kdy má server data k odeslání, ale klient je již nevyžaduje, a ukončí spojení pomocí příznaku `RST`. Příznak `RST` se používá taky ve chvíli, kdy klient neobrdží všechna data a žádá o znovu poslání. Bohužel TCP příznaky nám takové rozlišení neumožňují. Vždy musí vyplynout z kontextu komunikace. [2]

2 HTTPx – HTTP/1.0, 1.1, 2, 3

S rozvojem rychlé a efektivní síťové infrastruktury vznikla potřeba vývoje nových robustních protokolů, zejména aplikační vrstvy modelu TCP/IP, které dokáží naplno využívat služeb protokolů svých podvrstev.

Zásadním protokolem aplikační vrstvy pro přenos HTML (Hypertext Markup Language) souborů je protokol HTTP, jenž je základním protokolem pro webovou komunikaci, kde hypertextové dokumenty poskytují odkazy (hyperlinky) k dalším zdrojům.

První verze protokolu vznikla v roce 1989 jako RFC (Request For Comments) 1945 (HTTP/1.0) protokol typu Výzva – Odpověď (Challenge – Response) v komunikačním modelu Klient – Server. Avšak postupem času dosáhl svým návrhem nejen výkonostních, ale i bezpečnostních limitů. Například protokol HTTP/1.0 podporuje zaslání pouze jednoho požadavku na jedno otevřené TCP spojení. Postupem času tento problém vyřešila nová verze HTTP/1.1, která umožnila posílat více požadavků přes jedno otevřené TCP spojení. [3]

2.1 Základní princip komunikace

Základním prvkem HTTPx komunikace je server, což je vlastně služba běžící na počítači zpřístupňujícím internetový obsah pod WWW (World Wide Web) adresou. Klient je tzv. User agent, který provádí HTTP požadavky na server. Mezi typické user agenty řadíme webový prohlížeč, mobilní aplikaci, prohlížeč zvuků, videa... HTTP zdroje dat lze identifikovat pomocí URL (Uniform Resource Locator) adresy.[4]

$$URL = protokol(http) : //jmeno : heslo@www.vutbr.cz : 80/?tag = health/\#head$$

2.2 HTTP/1.0

Jedná se o první protokol aplikační vrstvy modelu ISO/OSI, který definuje základní principy internetové komunikace. Mimo jiné zavádí terminologii, kterou se řídí všechny další odvozené protokoly. Např.:

- connection (spojení) – spojení mezi programy na úrovni transportní vrstvy,
- message (zpráva) – základní jednotka HTTP komunikace,
- request (požadavek) – HTTP zpráva typu požadavek,
- response (odpověď) – HTTP zpráva typu odpověď,
- user agent (klient) – HTTP klient, který vytváří požadavky na webserver.

Protokol HTTP/1.0 byl poměrně rychle nahrazen protokolem HTTP/1.1 pro své bezpečnostní slabiny, mezi které patří nepoužití šifrování dat. Uživatelské přístupové údaje jsou přenášeny v textové podobě a bez časové závislosti na spojení. To umožňuje zranitelnost na útoky typu snooping. Proto zachycená data mohou být kdykoliv replikována.

Další slabinou je samotný návrh protokolu, kdy pro každý požadavek je vytvořeno nové TCP spojení. Většina činností je velmi pomalá a zvětšují šířku pásma. Každé spojení je ukončeno ihned po přijeté odpovědi, viz obr. 2.2. [5]

Základní metody, které protokol HTTP/1.0 podporuje, jsou:

- **GET** - slouží k získávání dat na určité adrese webového serveru,
- **POST** - slouží k dopravení informací od uživatele k webovému serveru,
- **HEAD** - slouží k získávání informací z metadat obsažených v záhlaví.

2.3 HTTP/1.1

Protokol HTTP/1.1 je v současné době nejvíc používaným protokolem definovaným v RFC 2616. Vznikl v roce 1999 a jedním z autorů byl tvůrce internetu Tim Berners-Lee. [6] Hlavním rozdílem oproti protokolu HTTP/1.0 je, že u něj nedochází k ukončování spojení po každém přenosu. Tím se snižuje náročnost na šířku pásma a samotná komunikace se zrychluje.[1] Zároveň vzniká možnost posílat HTTP požadavky na jednom TCP spojení bez nutnosti čekat na odpověď. Tomuto se říká HTTP Pipelining, který má obrovský vliv na zrychlení komunikace a snížení latence. Nevýhodu je, že server posílá odpovědi ve stejném pořadí, ve kterém je přijal – FIFO (First In, First Out), viz obr. 2.2. [7]

V protokolu HTTP/1.1 se objevuje povinné záhlaví `HOST`, které udává cílovou destinaci paketu. Poskytuje příležitost využít různé mezilehlé prvky, např. IPS sondy nebo proxy servery, které od té doby umožňují vznik virtuálních hostů na webovém serveru.

S protokolem HTTP/1.1 přibyly následující metody.

- **PUT/DELETE** - Slouží ke smazání/vytvoření objektu na webovém serveru.
- **OPTIONS** - Slouží ke zjištění určitých informací o daném serveru. Nejčastěji se používá k dotazování webového serveru, které metody podporuje.
- **PATCH** - Slouží ke změně již existujícího zdroje na webovém serveru.
- **TRACE** - Slouží ke zjištění informací na cílovém bodu. Slouží pro debugování. Obdoba softwarového `traceroute`.
- **CONNECT** - Nejčastěji se používá k vytvoření tunelu přes HTTP proxy k zabezpečenému webovému serveru.

[1] Odpadá nutnost pokaždé provádět 3-way handshake.

Dále vznikla zabezpečená nádstavba HTTPS (Hypertext Transfer Protocol Secure), která přidává šifrovanou vrstvu SSL/TLS (Secure Socket Layer/Transport Layer Security). Klient má možnost ověřit totožnost webového serveru, čehož v současné době využívá 56,9 % celkové internetové komunikace.[8]

2.3.1 Bezpečnostní výzvy protokolu HTTP/1.1

Head-of-Line blocking – z pohledu HTTP/1.1 protokolu, každý klient (webový prohlížeč) má omezené množství spojení se serverem. Pokud tento počet vyčerpáme, tak následující požadavky budou vyčkávat na ukončení předchozích. Toho lze dosáhnout například tím, že vytvoříme maximální počet paralelních spojení, která nebudou ukončena v patřičné době. [9]

Protokol HTTP/2 tento problém řeší pomocí multiplexování HTTP/2 streamů (proudů) do jednoho TCP spojení.[2] U protokolu HTTP/2 však vyvstává ten stejný problém na úrovni transportní vrstvy, kde pokud se ztratí některý z paketů TCP komunikace, dosáhneme podobného stavu. Tento problém řeší až protokol QUIC – základ protokolu HTTP/3. [6] [10]

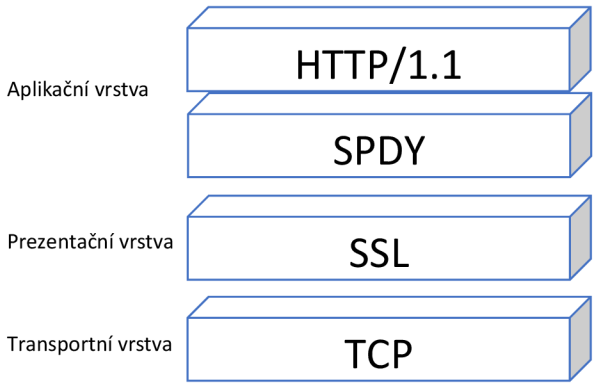
2.3.2 SPDY

SPDY („speedy“) je protokol aplikační vrstvy modelu ISO/OSI, obr. 2.1, vyvíjený společností Google, jehož cílem je zrychlení načítání webového obsahu. V laboratorních podmínkách je o 34 % rychlejší než protokol HTTP/1.1. [11]

Nevýhodou protokolu HTTP/1.1 je, že požadavky může inicializovat pouze klient. Server nemá žádnou možnost, jak klientovi poskytnout určitá data, o kterých ví, že je klient bude potřebovat. Dalším problémem je proměnlivá velikost záhlaví, která může mít velikost od jednotek B až po 2 kB. Komprese záhlaví na stanovenou délku urychlí komunikaci, sníží latenci a zrychlí serializaci požadavku. U protokolu HTTP/1.1 velmi často docházelo k odeslání stejného záhlaví přes jedno TCP spojení.

Mezi základní funkcionalitu protokolu SPDY patří multiplexování proudů, prioritizace požadavků a komprese HTTP záhlaví. Multiplexování proudů umožňuje vytvořit více konkurentních proudů přes jedno TCP spojení. Z toho plyne, že pro síťovou komunikaci je potřeba menší počet otevřených spojení. Prioritizace požadavků umožňuje klientům určit prioritu každému z nich. Tím pádem odpadá možné zahlcení ze strany serveru. Server v případě využití větší šířky pásma spojení posílá pouze pakety s největší prioritou. Komprese HTTP záhlaví snižuje velikost jednotlivých paketů.

[2]Multiplexování proudů je metoda, kdy je několik HTTP proudů kombinováno do jednoho TCP spojení. Obecně se jedná o kombinování analogových nebo digitálních signálů do jednoho signálu.



Obr. 2.1: SPDY protokol

Mezi rozšiřující funkce protokolu SPDY patří Server Push a Server Hint. Server Push umožňuje odeslat klientovi data, o kterých si myslí, že je bude klient potřebovat. Tato metoda urychluje vytvoření spojení klienta se serverem. Metoda Server Hint nabízí klientovi data, o kterých si myslí, že je bude klient potřebovat. Na rozdíl od Server Push, server očekává příznivou odpověď od klienta pro odeslání dat. Tato metoda vede taky ke zrychlení načítání dat.

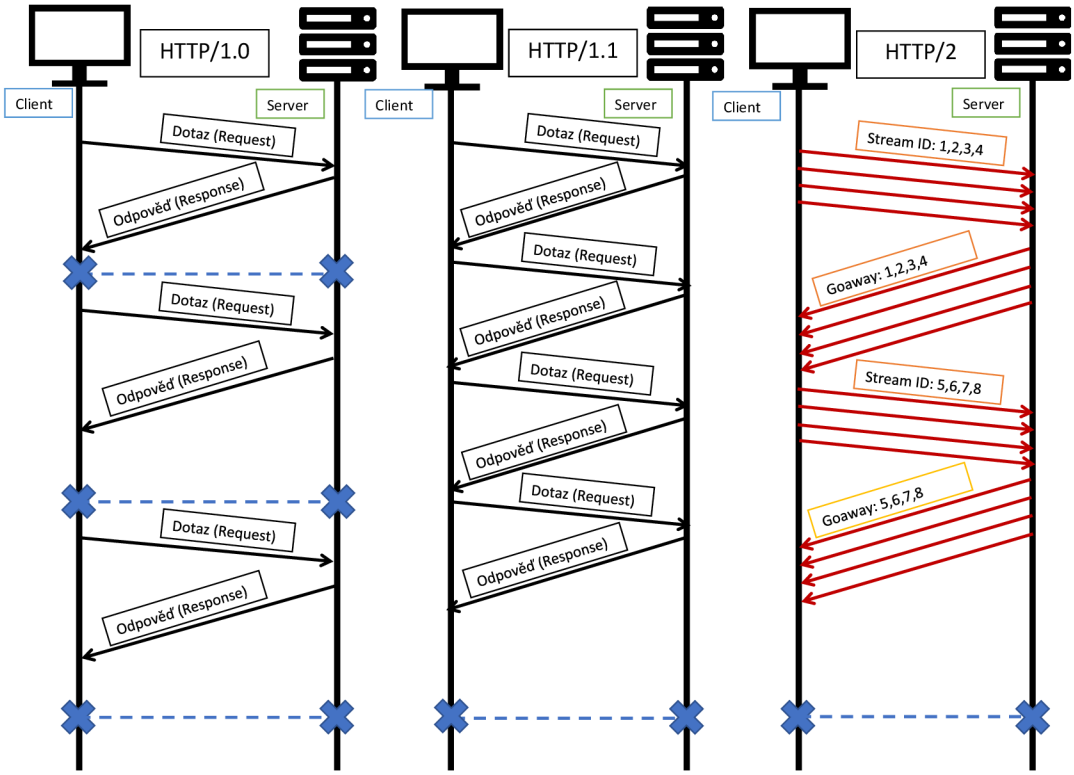
Vývoj tohoto protokolu byl ukončen v roce 2016, protože byl nahrazen protokolem HTTP/2, ve kterém byly funkcionality jako Server Push, multiplexování proudů, prioritizace požadavků nebo komprese záhlaví, uplatněny. [11] [12]

2.4 HTTP/2

S masivním používáním protokolů rodiny HTTP/1.x a rozvojem webových technologií začaly narůstat doby pro načtení webové stránky (Load Time). Proto vznikla potřeba efektivnějšího využívání protokolu TCP a tudíž vzniku nového aplikačního protokolu, který ovšem zachová veškerou funkcionalitu protokolu HTTP/1.1.[3] Avšak je kompatibilní pouze v jednom směru, od klienta směrem k serveru, pomocí metody Upgrade. Protokol HTTP/2 byl přijat organizací IETF v květnu 2015 jako RFC 7540.

Protokol HTTP/2 lépe a efektivněji využívá síťových prostředků se snížením latence. Toho je docíleno novým kompresním formátem záhlaví HPACK, paralelním zpracováním proudů v rámci jednoho spojení a prioritizaci požadavků, viz obr. 2.2. [13]

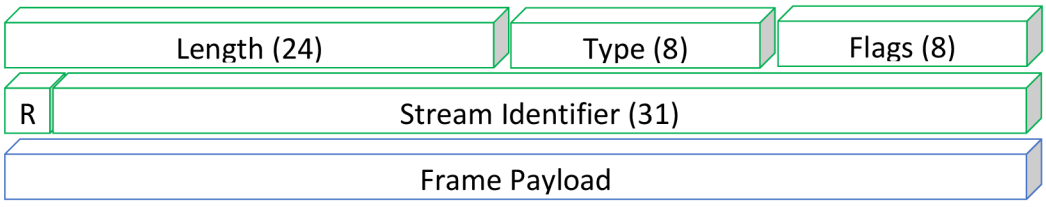
[3]Zachování HTTP metod, stavových kódů (status code), odpovědí a podoby záhlaví.



Obr. 2.2: Srovnání průběhu komunikace protokolů HTTP/1.0, 1.1, 2

2.4.1 Vlastnosti protokolu

Protokol HTTP/2 vychází svým návrhem z protokolu SPDY. Hlavním vylepšením je vrstva binárního kódování. Tato vrstva mění způsob přenosu dat. Základní nejmenší jednotkou HTTP/2 komunikace je rámec (frame).



Obr. 2.3: HTTP/2 rámec

Každý rámec obsahuje záhlaví rámce (frame header) a data (frame payload). Rámec záhlaví obsahuje svoji délku (Length), typ (Type), příznak (Flags), číslo proudu (Stream ID) a samotná data (Frame Payload), viz obr. 2.3. Kolekci rámců nazýváme zprávy (messages). Tyto zprávy vytvářejí kompletní požadavek nebo odpověď. Mezi

jednotlivé prvky rámce patří následující prvky.

- Length (délka) – Toto pole udává velikost rámce. Velikost může být sjednána při dohodnutí spojení mezi klientem a serverem.
- Type (typ) – Toto pole udává účel rámce. Dělíme jej na 10 podtypů.

  `HEADERS` – HTTP záhlaví dotazu nebo odpovědi klienta.

  `DATA` – Udává vlastní přenášená data.

  `PRIORTIY` – Specifikuje významnost proudu.

  `RST_STREAM` – Upozorňuje na chybu a ukončuje spojení.

  `SETTINGS` – Určuje vyjednávané vlastnosti spojení.

  `PUSH_PROMISE` – Oznamuje klientovi, že budou data odeslána bez vyžádání.

  `PING` – Slouží k otestování spojení.

  `GOAWAY` – Dává zprávu protistraně, aby neprodukovala další proudy pro vytvořené TCP spojení.

  `WINDOW_UPDATE` – Slouží k řízení toku dat.

  `CONTINUATION` – Slouží k pokračování sekvence záhlaví. Odeslán, pokud dojde k vyčerpání maximální velikosti záhlaví.
- Flags (příznak) – Jedná se o logickou hodnotu, která specifikuje informace o HTTP rámci. Pouze rámec typu `DATA` definuje příznak `END_STREAM`, nastaven na hodnotu 1, určuje konec proudu, a příznak `PADDED`, nastaven na hodnotu 1, indikuje přítomnost odsazení. Potom rámec typu `HEADERS` definuje `END_STREAM`, `PADDED`, `END_HEADERS`, nastaven na hodnotu 1, indikuje ukončení rámce záhlaví (headers frame), `PRIORITY`, nastaven na hodnotu 1, indikuje, že byla nastavena priorita proudu.
- Stream ID (číslo proudu) – Slouží k identifikaci určitých rámců k danému proudu. Jeden rámec může být členem pouze jednoho proudu. [14]

Jednotlivé rámce tvoří HTTP/2 proud, který je obousměrným spojením mezi klientem a serverem. Tento proud může přenášet jeden nebo několik zpráv s jedním nebo několika rámci. Vše předchozí je přenášeno přes jedno otevřené TCP spojení.

Mezi další vlastnosti HTTP/2 protokolu patří vytvoření paralelních požadavků a odpovědí (stream multiplexing). Umožňuje rozdělit zprávy do více nezávislých rámců, snížit nároky na šířku pásma a zrychlit komunikaci. A následně tyto rámce znovu složit v ucelenou zprávu. Tyto rámce jsou jednoznačně identifikovány pomocí čísla proudu.

Prioritizace proudů (stream prioritization) umožňuje přednostní přenos rámců s nastavenou prioritou. Pro rozlišení hodnoty priority, HTTP/2 protokol zavádí závislost proudu a váhu proudu. Každý proud může mít přiřazenou hodnotu priority od 1 do 256, ta může být přiřazena k jinému proudu pomocí čísla proudu. Kombinancí těchto parametrů vzniká datová struktura strom priority. [15] [16] [17]

2.4.2 Kontrola průběhu spojení

Kontrola průběhu spojení (Flow Control) je mechanismus ochrany příjemce před zahlcením daty. Jelikož HTTP/2 protokol má otevřeno pouze jedno TCP spojení s klientem, nelze využít řízení provozu (Flow Control) transportní vrstvy. Protokol HTTP/2 poskytuje základní stavební bloky na řízení spojení:[13] [17] [18]

- Každý příjemce může určit velikost okna (window size), které je schopen přijmout. Tato velikost může být specifikována na úrovni celého spojení nebo jednotlivého proudu.
- Každý příjemce vytvoří úvodní spojení (initial connection) a okno řízení proudu (velikost je v bajtech). Každý průchod rámce typu `DATA` snižuje velikost tohoto okna a naopak každý rámec typu `WINDOW_UPDATE` zvyšuje.
- Po uskutečnění spojení si klient a server vyměnují rámce typu `SETTINGS`, které nastaví výchozí velikost okna řízení provozu. Výchozí hodnota je nastavena na 65535, ale odesílatel ji může upravit na maximálních 2147483647. K udržování této hodnoty slouží odesílaní rámce typu `WINDOW_UPDATE`.
- Metoda řízení provozu není aplikovaná jen mezi koncovými stanicemi, nýbrž i mezi všemi mezilehlými prvky.

2.4.3 Vynucený přenos dat

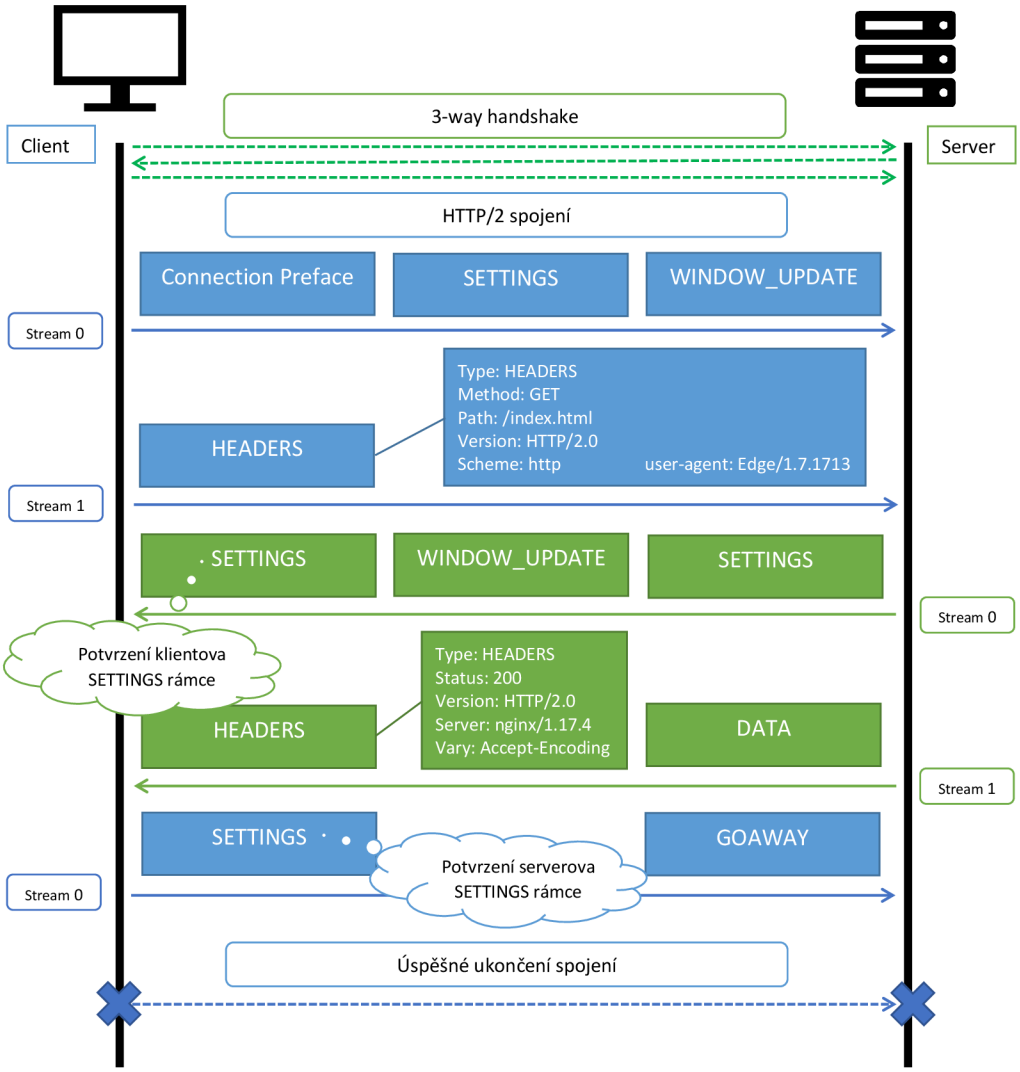
Typem vynuceného přenosu dat je metoda Server Push. Slouží přenosu dat ze strany serveru ke klientovi. Tato data nejsou klientem vyžádána, ale server usoudí, že je bude klient potřebovat. Především se jedná o obrazová data, stránky, kaskádové styly (CSS). Tyto zdroje mohou být načítány do mezipaměti v mezilehlých prvcích a znovu použity na různých stránkách. Platí zde prioritita ze strany serveru. Výsledkem je rychlejší doba načítání webové stránky. [18]

Server Push je inicializován pomocí rámce typu `PUSH_PROMISE`, který má za cíl oznámit klientovi, že mu budou odeslána nová data. Klient reaguje rámcem typu `RST_STREAM` pouze, pokud nemá zájem o data. Klient může serveru zakázat použití tohoto mechanismu pomocí parametru spojení `SETTINGS_ENABLE_PUSH`.

2.4.4 HPACK

Každé HTTP spojení obsahuje záhlaví, které může být v některých případech větší než samotná data. Proto vznikla potřeba vytvořit kompresní mechanismus. Předchozí protokol SPDY používá pro kompresi, dnes už zranitelný, algoritmus ZLIB. Proto byl vyvinut nový protokol HPACK.

Umožňuje zakódovat záhlaví pomocí statického Huffmanova kódu, který redukuje jeho velikost. Klient a server musí udržovat tabulku záhlaví (header field table)



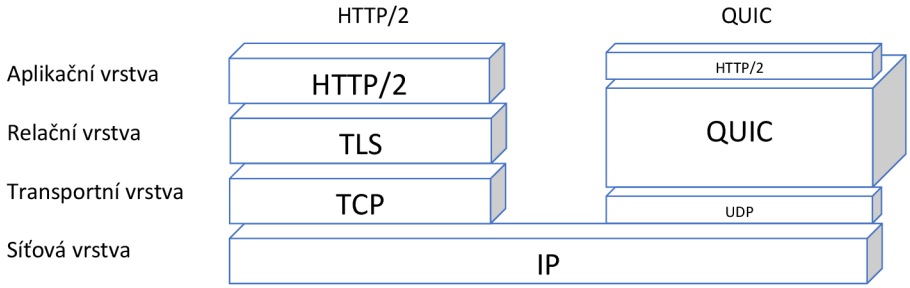
Obr. 2.4: HTTP/2 komunikace

zvlášť pro každý směr provozu. Tyto tabulky platí pro celé TCP spojení. Huffmanovo kódování umožňuje kopresi individuálních hodnot. Pro přenos stejných hodnot se použije pouze index záhlaví, ve kterém již byly použity. HPACK poskytuje dva druhy tabulek, statickou a dynamickou tabulku. Statická tabulka je definovaná před začátkem spojení, kdežto dynamická tabulka je vyplňovaná v průběhu síťového provozu.[19]

2.5 QUIC

QUIC (Quick UDP Internet Connections) je protokol transportní vrstvy vyvíjený firmou Google. Tento protokol je postaven nad protokolem UDP, viz obr. 2.5.[4] Nelze jej přesně zařadit do jediné vrstvy modelu ISO/OSI. Implementuje některé vlastnosti transportní vrstvy, jako je ochrana zahlcení, číslování rámců a zajištění doručení všech dat. Z relační vrstvy implementuje samotné šifrování dat. A z aplikační vrstvy využívá kompresi dat a komunikaci s protokolem HTTP/2.

Tento protokol kombinuje typický TCP 3-way handshake a TLS handshake. Tím je dosaženo zjednodušení vytváření spojení, šifrování dat již od zahájení spojení a autentizace všech účastníků.[20]

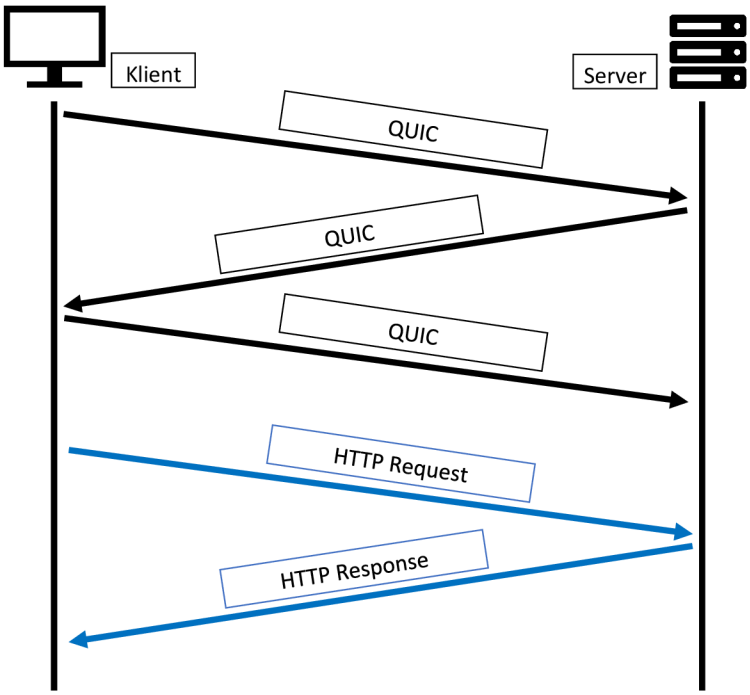


Obr. 2.5: QUIC protokol

Pro sestavení spojení protokolem QUIC je potřeba jednu až dvě zprávy, viz obr. 2.6. Oproti tomu předchozí protokol TCP použije 3 zprávy pro handshake a další 3 zprávy pro inicializaci zabezpečeného spojení TLS (TLS ClientHello, TLS ServerHello, TLS Finished). Pokud klient sestavuje spojení poprvé, zašle nešifrovanou zprávu pro vytvoření spojení (CHLO) a server odpoví zamítající zprávou (REJ), která obsahuje konfiguraci serveru, veřejný certifikát serveru a token zdrojové adresy serveru (Source-Address Token). [10]

Při každém dalším navazování spojení s tímto serverem, už může klient použít uložené údaje. Pro navázání dalšího spojení stačí použít zašifrovanou zahajovací zprávu.[20]

[4]Protokol UDP nezaručuje správné pořadí doručení paketů, ani ověření jejich správného doručení.



Obr. 2.6: HTTP požadavek nad protokolem QUIC

2.6 HTTP/3

V listopadu roku 2018 byl vytvořen návrh protokolu HTTP/3, který staví na již zmiňovaných protokolech QUIC a HTTP/2. Jedná se o binární protokol přenášený QUIC protokolem. Významnou vlastností tohoto protokolu je, že všechna data jsou šifrovaná. A každý proud je nezávislý na jiném.

Jakmile klient zjistí, že cílový server umožňuje komunikaci protokolem HTTP/3, vytvoří QUIC spojení se serverem. Tento protokol umožňuje sestavení spojení, šifrování dat, multiplexování proudů a kontrolu provozu. Základní jednotkou je rámec. Stejně jako u protokolu HTTP/2, slouží rámce (DATA, HEADERS) k vytvoření základních požadavků a odpovědí.

Jednotlivé proudy umožňují multiplexování požadavků, kde jeden pár požadavek – odpověď odpovídá jednomu QUIC proudu. Tyto proudy jsou nezávislé na sobě a každý je, na rozdíl od protokolu HTTP/2, zvlášť šifrovaný. Další metodou z protokolu HTTP/2 je Server Push. U protokolu HTTP/3 přibyly řídící metody, které omezují vlastnosti metody Server Push.

Záhlaví paketů je kódováno. Oproti protokolu HTTP/2, kde se používá metoda HPACK, je zde použita metoda QPACK, která jej nahrazuje. [21] [22]

3 Útoky DoS a DDoS

Útoky typu DoS (Denial of Service), nebo DDoS (Distributed Denial of Service) jsou útoky typu odepření služby pro ostatní uživatele. Jedná se buď o útoky využívající chyby daného zařízení, nebo o útoky generující síťový provoz zahlcující oběť větším počtem požadavků. Tyto útoky většinou cílí na webové servery, ale i na databázové servery, Domain Name System (DNS) servery, servery emailové komunikace nebo servery poskytující data. Cílem těchto útoků je znepřístupnit server oběti, zvýšit latenci odpovědí na požadavky klientů, snížit celkovou kvalitu spojení nebo neumožnění se připojit k cílovému serveru.

Základní dělení DoS útoků je na útoky využívající zranitelnost systému, zejména cílící na přetečení (Buffer Overflow Attacks), kde tento typ útoku vyčerpá veškerou dostupnou paměť, případně veškerý procesorový čas, nebo místo na pevném disku. Většinou k provedení stačí malý počet paketů. Tento typ útoku vede k nepredikovatelnému chování systému, až k celkové nedostupnosti.[23]

Druhým typem jsou záplavové útoky (Flood Attacks), které cílí na webový server velkým počtem vytvořených spojení. Pokud cílový server nemá patřičnou ochranu, stane se nedostupným pro další požadavky.[1]

Dalším typem jsou útoky cílící na chybnou či nedokonalou implementaci protokolu. Čehož lze poměrně snadno využít vytvořením neúplného nebo pozměněného spojení a tím způsobit nekontrolovaně otevřené spojení. Pokud takových spojení vytvoří útočník víc, dojde k vyčerpání dotazů a následné nedostupnosti serveru.

Musíme rozlišovat, na které vrstvě protokolu TCP/IP se daný útok nachází. DDoS útoky probíhají nejčastěji na transportní vrstvě. A na vyšších vrstvách, konkrétně na aplikační vrstvě útoky typu DoS.

3.1 Záplavové útoky transportní vrstvy

Tyto útoky využívají protokolu TCP nebo UDP. Útočník se snaží vyčerpat šířku pásma oběti. Jedná se například o útok UDP Flood nebo útoky využívající zranitelnosti v protokolech transportní vrstvy (TCP SYN Flood).

Tyto útoky jsou relativně snadno detekovatelné, protože se většinou jedná o útoky s velkým síťovým provozem.[25]

[1] Aby tyto útoky byly úspěšné, musí mít útočník větší kapacitu na tvorbu nových spojení, než má samotný napadený server.

3.2 Záplavové útoky aplikační vrstvy

Útoky na aplikační vrstvě nevyužívají takového síťového provozu jako útoky na transportní vrstvě. Tyto typy útoků mají za úkol zahltit oběť validními požadavky a docílit nedostupnost. Řadíme zde VOIP Flooding, DHCP Starvation Attack, DNS Amplification Attack.

3.3 Pomalé DoS útoky

Pomalé DoS útoky (Slow DoS) kombinují typ záplavového útoku a útoku na zranitelnost. Většinou cílí na aplikační vrstvu modelu ISO/OSI.

Tyto útoky vytváří pouze omezený počet nekompletních požadavků. Tyto požadavky je velmi obtížné odlišit od normálního síťového provozu. Proto lze tento útok špatně detekovat detektory DoS útoků.

Útočník vytváří několik paralelních TCP spojení, ve kterých posílá tyto požadavky. Napadený server nechává otevřená veškerá spojení s útočníkem, i když v nich neprobíhá žádný přenos dat. Pokud je veškeré dostupné místo určené pro příjem síťové komunikace vyčerpané, server není schopen přijímat další požadavky. Tyto útoky se dají zefektivnit tím, že útočník vypozoruje chování serveru na modifikované spojení, např. dobu, po kterou nechá server spojení otevřené. A následně těsně před resetováním serveru pošle modifikované spojení znovu, tak aby vytvořené spojení zůstalo otevřené po celou dobu útoku.

Pomalé DoS útoky dělíme na [24]

- DoS útoky s čekajícími požadavky
- DoS útoky s dlouho trvající odpovědí
- vícevrstvé DoS útoky

Útoky s čekajícími požadavky využívají možnost vkládat data u HTTP/2 protokolu do více rámců (např. rámec typu `CONTINUATION`) a tím pádem odeslat dotaz v nekompletní podobě. Tato vlastnost sama o sobě nestačí na úspěšné provedení útoku. Ten je možno provést v případě, že protokol HTTP/2 bude mít chybu v implementaci. Server alokuje zdroje pro nekompletní dotaz a útočník se snaží vyčerpat veškeré zbývající zdroje serveru.

Útoky s dlouhotrvající odpovědí se vyznačují odesláním validního HTTP požadavku odeslaného útočníkem na server. Pomocí speciálně zvolených parametrů je možno donutit server posílat odpověď po malých blocích. Server posílá velmi málo dat směrem k útočníkovi, ale spojení zůstává pořád otevřené. Při provedení několika podobných spojení, vyčerpá server všechny zdroje a nebude schopen obsloužit validního klienta.

4 Slow DoS útoky na protokol HTTP/2

S příchodem nového hypertextového protokolu vyvstává otázka bezpečnosti. HTTP/2 spojení může být, i přes kompresi hlaviček, náročnější na paměťové místo. Velikost alokovaného místa vymezují rámce `SETTINGS`. Právě tento rámec může být zneužit pro útok typu Slow DoS záměnou, opakováním jednotlivých částí rámce nebo tvorbou nových nedefinovaných parametrů. Taktéž lze zneužít rámce `WINDOW_UPDATE` a `PRIORITY` podobným způsobem.

Další možností je zahltit cílový bod velkým počtem malých prázdných rámců a donutit jej prodloužit dobu pro zpracování.[1][13]

Speciálně vytvořené HTTP/2 proudy cílí na volné síťové prostředky daného serveru. Tyto útoky se vesměs skládají ze dvou částí. Nejprve se snaží, vhodně zvolenými rámci, docílit vytížení síťového procesu, případně celého vlákna na co nejdelší dobu. A následně znepřístupnit cílový server běžným požadavkům tím, že vyčerpají veškerou zbývající kapacitu, ať už procesorových vláken, nebo paměťového prostoru určeného pro síťové procesy v paměti RAM. Oproti protokolu HTTP/1.x, kde by útočník otevíral tolik TCP spojení, kolik by dokázal napadený server obsloužit, u HTTP/2 útočník využívá multiplexování většího počtu proudů na jedno TCP spojení. [26]

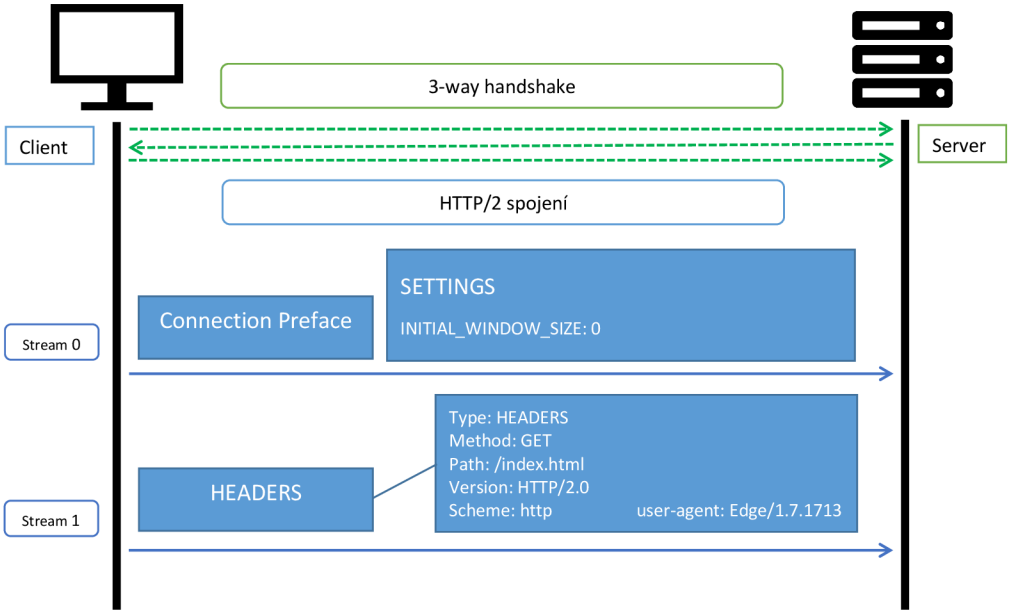
Server Microsoft IIS 10.0 nepodporuje nešifrovanou podobu protokolu HTTP/2 – h2c. Protokol h2c (hypertext-to-cleartext) je implementován nad TCP protokolem, kdežto h2 pracuje nad TLS protokolem. Proto níže implementované Slow DoS útoky nebudou pro IIS 10.0 fungovat.[25] [27]

4.1 Útok Slow READ

K vytvoření útoku Slow READ, musí útočník nastavit rámci `SETTINGS` příznak `SETTING_INITIAL_WINDOW_SIZE` na hodnotu 0 a vytvořit kompletní požadavek typu GET, jak je znázorněno na obr. 4.1. Hodnota `SETTING_INITIAL_WINDOW_SIZE: 0` říká serveru, že klient je momentálně zaneprázdněn a tím pádem bude očekávat rámec `WINDOW_UPDATE`, který ovšem útočník nikdy nepošle. Tím pádem dojde k zablokování daného proudu a server bude vyčkávat po dobu, kterou má specifikovanou ve svém nastavení jako TIMEOUT. [2]

[1] Pozor rámce typu DATA nebo CONTINUATION lze posílat na konci proudu prázdné zcela validně.

[2] TIMEOUT - Doba, po jejíž uplynutí server uzavře spojení.



Obr. 4.1: Znázornění útoku Slow READ

4.1.1 Průběh útoku pro jednotlivé webservery

Servery Apache 2.4.17, 2.4.18 jsou zranitelné na útok Slow READ, protože dedikují jedno procesorové vlákno na jeden proud.[28] Server udržuje otevřené pouze jedno TCP spojení. Avšak pomocí metody multiplexování proudů může přes toto spojení přenést více HTTP/2 proudů. Od verze Apache 2.4.20 je jedno vlákno určeno pro jedno TCP spojení. A jedno TCP spojení může obsloužit n proudů. Pro úspěšný útok musí útočník vytvořit několik TCP spojení obsahující alespoň jeden HTTP/2 proud.[25]

Spojeni se serverem Apache 2.4.17 nebylo uzavřeno ani po 10 minutách. Vyčkává na rámec `WINDOW_UPDATE`, který ovšem nikdy nedostane. Oproti tomu Apache 2.4.29 uzavře spojení po 300 sekundách a Apache 2.4.41 po 60 sekundách, oba s rámci `GOAWAY` s kódem chyby `NO_ERROR (0)`. Webový server Nginx 1.14.0 uzavře spojení po 240 sekundách s rámcem `RST_STREAM` s chybovou hláškou `PROTOCOL_ERROR (1)` a následně s rámcem `GOAWAY` s kódem chyby `NO_ERROR (0)`. (tab. 4.1)

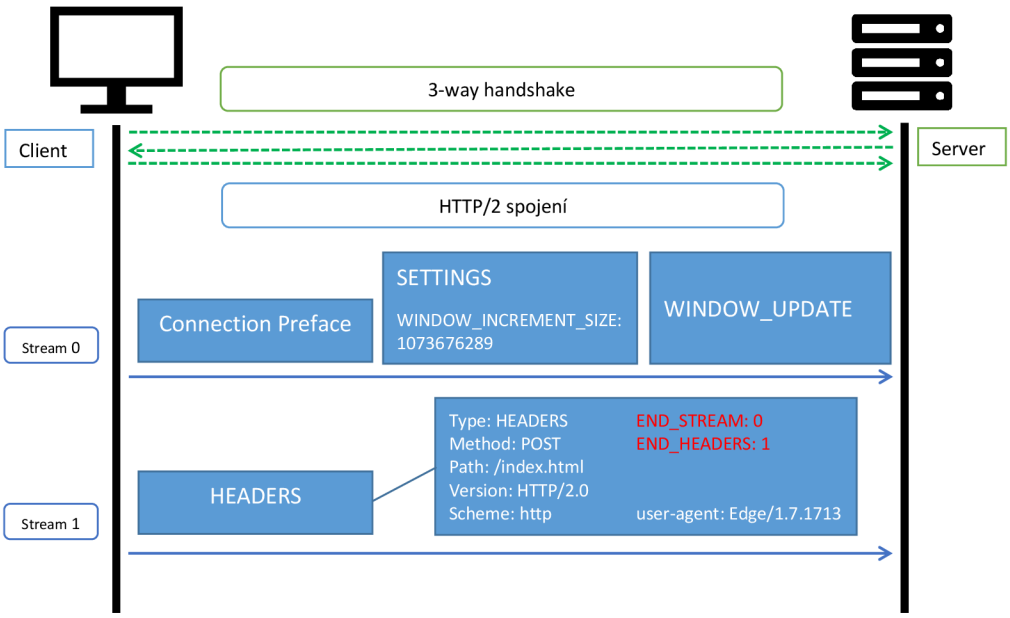
Pokud známe jednotlivé doby, po které zůstane spojení otevřené, stačí vytvořit dostatečný počet paralelních spojení a tím zahltit všechny dostupné prostředky serveru. Na útok Slow READ je zranitelný pouze webový server Apache 2.4.17, viz tab. 4.2. [25] Výsledek průběhu útoku Slow READ na server Apache 2.4.17 s použitím generátoru `slowhttp2test.py` z kapitoly 6 je zobrazen na grafu B.1.

4.2 Útok Slow POST

Tento útok se zaměřuje na metodu POST. Jak už bylo zmíněno, tato metoda se nachází v rámci `HEADERS`. Tento rámec mimo jiné obsahuje další čtyři příznaky: `END_STREAM, END_HEADERS, PADDED, PRIORITY`. Pro úspěšný útok je potřeba využít příznak `END_STREAM, END_HEADERS`: [25]

- `END_STREAM` - Tento příznak má velikost 1b (0x1) a pokud je nastaven na hodnotu 1, tak oznamuje, že se jedná o poslední blok hlaviček, který cílový bod odešle na daném čísle proudu. Pokud je nastaven na hodnotu 0, tak udává, že odesílatel má stále rámec typu `DATA` k odeslání.
- `END_HEADERS` - Tento příznak má velikost 1b (0x4) a pokud je nastaven na hodnotu 1, tak oznamuje, že obsahuje celý blok hlaviček a tudíž nebude následovat žádný další rámec typu `CONTINUATION`. Pokud je nastaven na hodnotu 0, musí následovat rámec `CONTINUATION`. Cokoli jiného musí být označeno chybou spojení - `PROTOCOL_ERROR`.[13]

Pro úspěšný útok musí útočník nastavit příznaky rámce `HEADERS` na následující hodnoty `END_STREAM: 1, END_HEADERS: 0` a zároveň poslat validní metodu POST (obr. 4.2). Tím dosáhne toho, že server bude čekat na další datový rámec `DATA` po omezenou dobu.



Obr. 4.2: Slow POST

4.2.1 Průběh útoku pro jednotlivé webservery

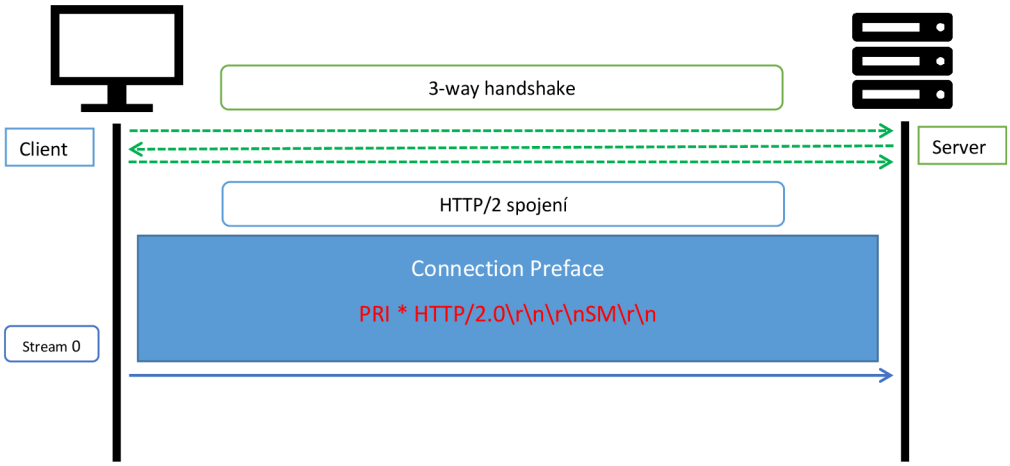
Server Apache 2.4.17 spojení nikdy neuzavře. Oproti tomu Apache 2.4.29 uzavře spojení po 5 sekundách a Apache 2.4.41 po 5 sekundách. Oba s rámci `RST_STREAM` bez chybové hlášky a `GOAWAY` s kódem chyby `NO_ERROR (0)`. Nginx 1.14.0 uzavře spojení po 180 sekundách s rámcem `GOAWAY` a kódem chyby `NO_ERROR (0)`, viz tab. 4.1.

Na útok Slow POST je zranitelný webový server Apache 2.4.17 (tab. 4.3). K úspěšnému útoku je potřeba vytvořit maximální počet spojení - 400 spojení, viz tab. 4.2.

Výsledek průběhu útoku Slow POST na server Apache 2.4.17 s použitím generátoru `slowhttp2test.py` z kapitoly 6 je zobrazen na grafu B.2.

4.3 Útok Slow PREFACE

Pro zahájení komunikace protokolem HTTP/2 je po úspěšném vyjednání TCP spojení[3] poslán klientem tzv. `MAGIC` rámec nebo také `CONNECTION PREFACE` rámec. Tento rámec má následující tvar `"PRI * HTTP/2.0\r\n\r\nSM\r\n"` (obr. 4.3).



Obr. 4.3: Slow PREFACE útok

Server po přijetí tohoto paketu jednak očekává potvrzení vyjednávaných parametrů spojení - rámec `SETTINGS`, tak i samotný validní HTTP/2 požadavek. Útočník tento požadavek nikdy nepošle a donutí server čekat, dokud server samotný spojení neuzavře.[25]

[3] 3-Way Handshake – viz 1.1

4.3.1 Průběh útoku pro jednotlivé webservery

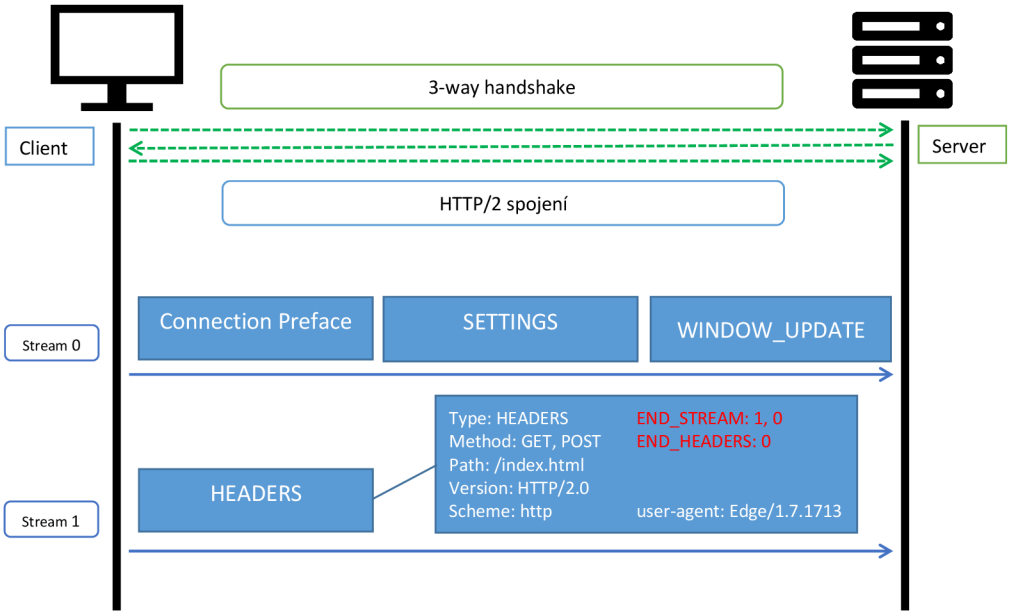
Server Apache 2.4.17 uzavře spojení na úrovni transportní vrstvy po 60 sekundách s pakety `FIN` a `ACK`. Apache 2.4.29 uzavře spojení po 300 sekundách a Apache 2.4.41 po 60 sekundách s rámcem `GOAWAY` a kódem chyby `NO_ERROR (0)`. Server Nginx 1.14.0 ukončí spojení po 30 sekundách s rámcem `GOAWAY` a kódem chyby `PROTOCOL_ERROR (1)`, viz tab. 4.1.

Na útok Slow PREFACE jsou zranitelné všechny testované servery Apache (tab. 4.2). K vytvoření úspěšného útoku je potřeba vytvořit maximální počet spojení. Pro server Apache 2.4.17 otevřeme 400 spojení, Apache 2.4.29 - 150 spojení a pro Apache 2.4.41 - 400 spojení, viz tab. 4.2. Takto vygenerovaná spojení způsobí nedostupnost, pro server Apache 2.4.17 a Apache 2.4.41 po dobu 60 sekund a pro server Apache 2.4.29 po dobu 300 sekund. Poté se spojení obnoví.

Výsledek průběhu útoku Slow PREFACE na server Apache 2.4.17 s použitím generátoru `slowhttp2test.py` z kapitoly 6 je zobrazen na grafu B.3.

4.4 Útok Slow HEADERS

Tento útok lze provést ve dvou provedeních. Buď požadavkem bude metoda GET nebo POST. Útočník, obdobně jako u útoku Slow POST, využije stejných příznaků `HEADERS` rámce. Konkrétně pro metodu GET, útočník nastaví `END_HEADERS: 0` a `END_STREAM: 1`. Příznak `END_HEADERS: 0` nám říká, že rámec `HEADERS` není kompletní a musí následovat další rámec `CONTINUATION`. Oproti tomu `END_STREAM: 1` označuje konec validního HTTP/2 proudu. Server bude čekat na rámec `CONTINUATION` s dalšími daty, avšak ty mu útočník nikdy neodešle. Po určité době server spojení ukončí. Pro útok s metodu POST útočník pošle rámec `HEADERS` s příznakem `END_HEADERS: 0` a `END_STREAM: 0` (obr. 4.4).[25]



Obr. 4.4: Slow HEADERS útok

4.4.1 Průběh útoku pro jednotlivé webservery

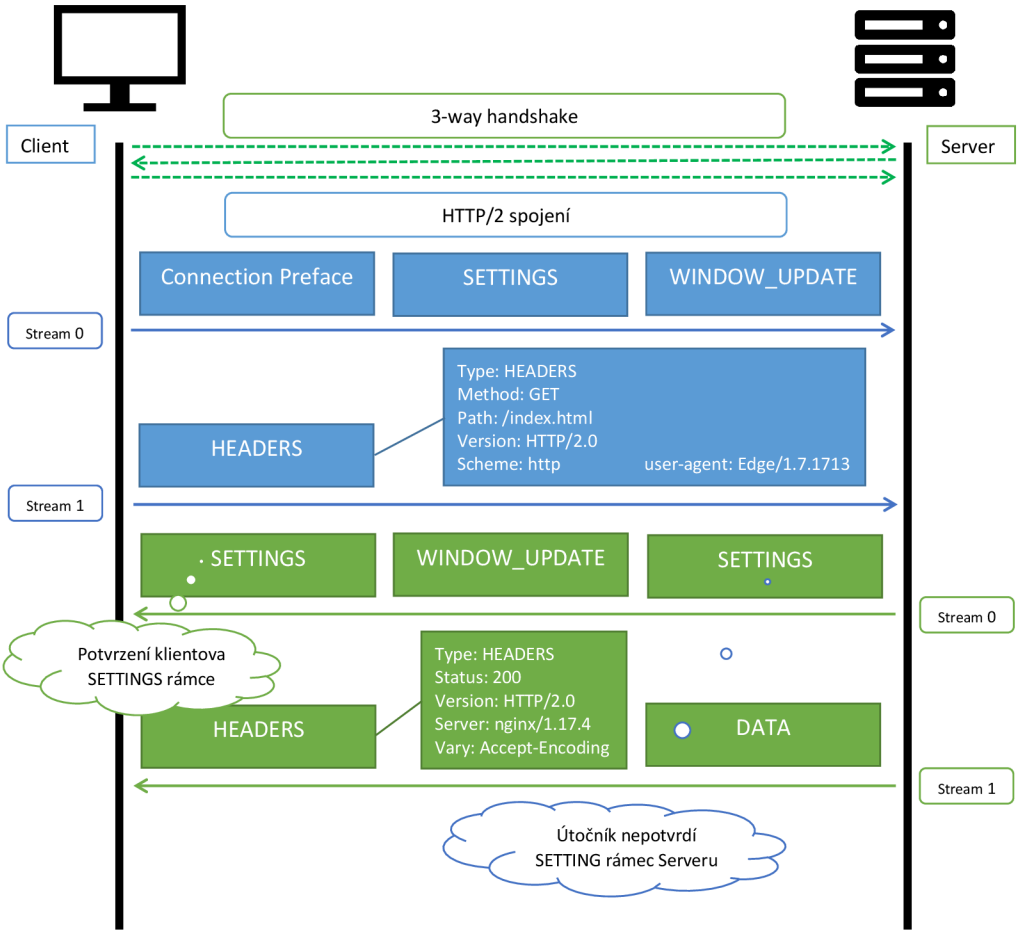
Server Apache 2.4.17 nikdy spojení neuzavře. Apache 2.4.29 uzavře spojení po 300 sekundách s rámcem `GOAWAY` a kódem chyby `NO_ERROR (0)`. Apache 2.4.29 pro obě metody uzavře spojení po 60 sekundách s rámcem `GOAWAY` a kódem chyby `NO_ERROR (0)`. Nginx 1.14.0 pro obě metody uzavře spojení po 90 sekundách s rámcem `GOAWAY` a kódem chyby `PROTOCOL_ERROR (1)`, tab. 4.1.

Výsledek průběhu útoku Slow PREFACE na server Apache 2.4.17 s použitím generátoru `slowhttp2test.py` z kapitoly 6 je zobrazen na grafu B.4.

4.5 Útok Slow SETTINGS

Poslední z útoků cílí na rámec SETTINGS. Tento rámec musí být, dle RFC 7540, potvrzen protistranou rámcem `SETTINGS: 0`. Pokud ovšem cílový bod neobdrží tento potvrzovací rámec, bude vyčkávat určitou dobu a poté uzavře spojení s chybou typu `SETTINGS_TIMEOUT`. [13]

Útočník pošle validní požadavek s metodou buď GET nebo POST. Server odpoví potvrzením SETTINGS rámce klienta a svým SETTINGS rámcem, na který bohužel od klienta neobdrží žádnou odpověď (obr. 4.5).[25]



Obr. 4.5: Slow SETTINGS útok

4.5.1 Průběh útoku pro jednotlivé webservery

Apache 2.4.17 uzavře spojení po 60 sekundách na úrovni transportní vrsty s pakety `FIN` a `ACK`. Apache 2.4.29 uzavře spojení po 5 sekundách s rámcem `GOAWAY` a kó-

dem chyby `NO_ERROR (0)`. Apache 2.4.41 uzavře spojení po 5 sekundách s rámcem `GOAWAY` a kódem chyby `NO_ERROR (0)`. Nginx 1.14.0 uzavře spojení po 180 sekundách s rámcem `GOAWAY` a kódem chyby `NO_ERROR (0)`, viz. tab. 4.1.

Na útok Slow SETTINGS je zranitelný pouze webový server Apache 2.4.17, viz tab. 4.2. K vytvoření úspěšného útoku je potřeba využít maxímálního počtu 400 spojení, viz tab. 4.1.

Výsledek průběhu útoku Slow PREFACE na server Apache 2.4.17 s použitím generátoru `slowhttp2test.py` z kapitoly 6 je zobrazen na grafu B.5.

4.6 Srovnání útoků

Pro úspěšné provedení útoků je potřeba znát maximální počet paralelních TCP spojení, které dokáže daný server současně obsloužit. Aby útočník dosáhl efektivního útoku, musí otevřít určitý počet TCP spojení, viz tab. 4.2. Taky je potřeba pracovat s jednotlivou dobou odezvy daného webového serveru pro jednotlivé útoky, viz tab. 4.1. Hodnoty byly zjištěny prováděním jednotlivých útoků na webové servery a měřením doby (programem `slowhttp2test.py`), kdy dojde k uzavření otevřeného spojení.

Server/Útok	S_READ	S_POST	S_PREFACE	S_HEADERS	S_SETT
Apache 2.4.17	∞	∞	60 s	∞	60 s
Apache 2.4.29	300 s	5 s	300 s	300 s	5 s
Apache 2.4.41	60 s	5 s	60 s	60 s	5 s
Nginx 1.14.0	240 s	180 s	30 s	90 s	180 s
IIS 10.0	x	x	x	x	x

Tab. 4.1: Doba, po které webové servery uzavřou spojení v závislosti na typu útoku

4.6.1 Maximální počet paralelních spojení

Webový server byl instalován ze standarního balíčkovacího systému APT s výchozí konfigurací pro víceprocesorový modul MPM (Multi-Processing Module) – `mpm_event_module`, který se nachází v konfiguračním souboru: `/etc/apache2/mods-enabled/mpm_event.conf`.[30] Odtud je možno zjistit maximální počet spojení - `MaxRequestWorkes 150`. Tento modul se načítá pomocí příkazu
`LoadModule mpm_event_module /usr/lib/apache2/modules/mod_mpm_event.so`.

Výpis 4.1: Maximální počet otevřených TCP spojení serveru Apache 2.4.29

```
<IfModule mpm_event_module>
StartServers              2
MinSpareThreads           25
MaxSpareThreads           75
ThreadLimit               64
ThreadsPerChild           25
MaxRequestWorkers         150
MaxConnectionsPerChild    0
</IfModule>
```

Obdobně lze zjistit maximalní počet spojení pro server Apache 2.4.17 a Apache 2.4.41 s jediným rozdílem, že tyto servery jsou zkompilované ručně. Nejprve je potřeba zjistit výchozí nastavení multiprocesorového modulu.[31] [32]

Výpis 4.2: Výchozí MPM modul pro servery Apache 2.4.17 a 2.4.41

```
#!/bin/bash
/usr/local/apache2/bin/apachectl -V | grep -i mpm
```

Dále lze zjistit, že server Apache 2.4.17 byl zkompilovám s výchozím nastavením `mpm_event_module`. V konfiguračním souboru
`/usr/local/apache2/conf/extra/httpd-mpm.conf` lze najít výchozí počet spojení pro daný multiprocesorový modul. Maximální počet paralelních spojení je – `MaxRequestWorkers  400`.

Výpis 4.3: Maximální počet otevřených TCP spojení serveru Apache 2.4.17

```
<IfModule mpm_event_module>
StartServers              3
MinSpareThreads           75
MaxSpareThreads           250
ThreadsPerChild           25
MaxRequestWorkers         400
MaxConnectionsPerChild    0
</IfModule>
```

Server Apache 2.4.41 byl zkompilován s výchozím nastavením `mpm_event_module`. V konfiguračním souboru `/usr/local/apache2/conf/extra/httpd-mpm.conf` pro daný multiprocesorový modul je uveden maximální počet paralelních spojení – `MaxRequestWorkers 400`.

Výpis 4.4: Max. počet otevřených paralelních TCP spojení – Apache 2.4.41

```
<IfModule mpm_event_module>
StartServers              3
MinSpareThreads           75
MaxSpareThreads           250
ThreadsPerChild           25
MaxRequestWorkers         400
MaxConnectionsPerChild    0
</IfModule>
```

Server Nginx 1.14.0 má ve výchozím nastavení počet spojení `worker_connections 768`.

Výpis 4.5: Max. počet otevřených paralelních TCP spojení - Niginx 1.14.0

```
cat /etc/nginx/nginx.conf | grep -i worker_connections
```

V tabulce 4.2 je uveden maximální počet paralelních TCP spojení na jednotlivé webové servery.

Server	Počet spojení
Apache 2.4.17	400
Apache 2.4.29	150
Apache 2.4.41	400
Nginx 1.14.0	768
IIS 10.0	x

Tab. 4.2: Maximální počet paralelních spojení podle serverů

Pole `MaxRequestWorkers` se do verze Apache 2.3.13 jmenovalo `MaxClients` a udává, maximální počet spojení, který dokáže daný server zpracovat. [32]

4.6.2 Porovnání jednotlivých Slow DoS útoků

Verze/Útok	S_READ	S_POST	S_PREFACE	S_HEADERS	S_SETT
Apache 2.4.17	✓	✓	✓	✓	✓
Apache 2.4.29	✗	✗	✓	✗	✗
Apache 2.4.41	✗	✗	✓	✗	✗
Nginx 1.14.0	✗	✗	✗	✗	✗
IIS 10.0	✗	✗	✗	✗	✗

Tab. 4.3: Zranitelnost webových serverů na jednotlivé typy útoků

Z tabulky 4.3 je patrná úspěšnost jednotlivých útoků na různé druhy webových serverů. Jedinou zranitelnou verzí na všechny typy představených Slow DoS útoků je Apache 2.4.17.

5 Testovací prostředí

Přítomnost protokolu HTTP/2 mezi webovými aplikačními protokoly se odhaduje k 8. 12. 2019 na 42,3 % zařízení. [34] Existuje několik implementací podporujících HTTP/2. Ze statistik je nejvíce používaný webový server Apache (42,8 %), následuje Nginx (30,9 %), Cloudflare Server (12,0 %) a v neposlední řadě Microsoft IIS (8,0 %). [35]

V tabulce 5.1 je uveden přehled nejpoužívanějších typů serverů Apache 2.4.x.

Taktéž je potřeba rozlišovat jednotlivé verze webových serverů. Zajímavým údajem je rozdělení typů operačních systémů hostujících daný webový server. K 8. 12. 2019 70,8 % webových stránek používá operační systémy typu UNIX, oproti tomu zbylých 29,2 % připadá na operační systém WINDOWS. [36]

Verze webového serveru	Procentuální zastoupení
2.4.41	17,4 %
2.4.39	17,1 %
2.4.6	13,5 %
2.4.18	12,1 %
2.4.25	8,4 %
2.4.29	8,3 %
2.4.10	8,0 %
2.4.7	6,7 %

Tab. 5.1: Verze webového serveru Apache 2 [37]

5.1 Lokální sestavení

Testovací síť je vytvořena za pomocí virtualizační technologie VMWare Player ve verzi 15.5 na hostovském operačním systému Microsoft Windows 10 s procesorem Intel i7-4710HQ a operační pamětí 16 GB. Pro demonstraci útoků byly vybrány 3 různé webové servery ve své výchozí konfiguraci: (obr. 5.1)

- Apache ve verzi 2.4.29 (2.4.17, 2.4.41) - host: Ubuntu 18.04 LTS (2GB RAM, 2 jádra)
- Nginx ve verzi 1.17.4 - host: Ubuntu 18.04 LTS (2GB RAM, 2 jádra)
- IIS ve verzi 10.0 - host: Microsft Windows 2019 Server (2GB RAM, 2 jádra)

Každému zařízení byly přiřazeny 2 síťové karty. Jedna umožňující přípojení přes NAT do sítě internet a druhá pro komunikaci uvnitř interní sítě.

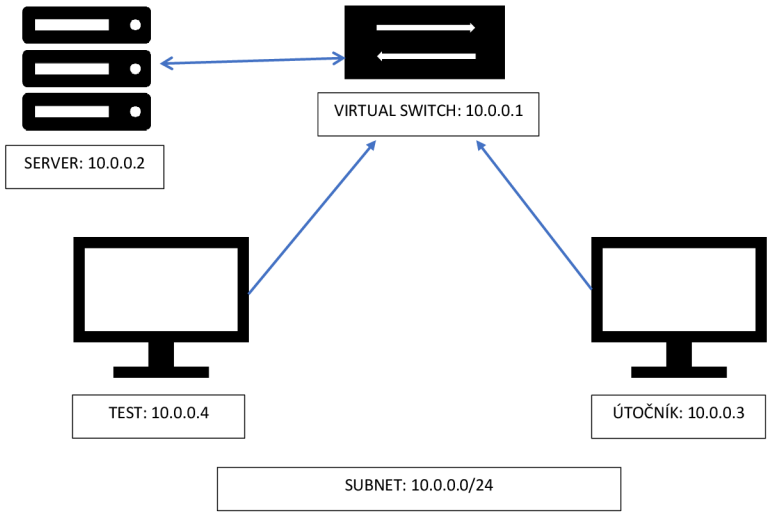
Virtualizační nástroj WMWare Player umožňuje tvorbu buď hostovské sítě (Host-only) nebo s překladem IP adres pomocí NAT. Pro alokaci lze zvolit automatické

přiřazování IP adres pomocí DHCP nebo je možné zadat jako statické. Avšak je potřeba dodržet dané konvence přiřazování IP adres, uvedené níže. [1]
IP adresy pro Host-only síť:

- <síť>.1 -> IP adresa hostovské stanice - virtuální switch,
- <síť>.2 - <síť>.99 -> Statické IP adresy,
- <síť>.100 - <síť>.253 -> IP adresy přiřazené DHCP serverem,
- <síť>.254 -> IP adresa DHCP serveru,
- <síť>.255 -> IP adresa sloužící pro všesměrové vysílání.

IP adresy pro NAT síť:

- <síť>.1 -> IP adresa hostovské stanice - virtuální switch,
- <síť>.2 -> IP adresa zařízení provádějící NAT,
- <síť>.3 - <síť>.99 -> Statické IP adresy,
- <síť>.100 - <síť>.253 -> IP adresy přiřazené DHCP serverem,
- <síť>.254 -> IP adresa DHCP serveru,
- <síť>.255 -> IP adresa sloužící pro všesměrové vysílání.



Obr. 5.1: Virtulizovaná testovací síť

[1] <síť> v našem případě odpovídá rozsahu 10.0.0.0/24

5.1.1 Virtuální síť

Server se od klientů liší aplikací webového serveru (obr. 5.1). Klient TEST slouží ke zjišťování dostupnosti webového serveru. 5.3 A ÚTOČNÍK je počítač generující útok. Na něm v kapitole 6 poběží generátor útoků `slowhttp2test.py`.

5.2 Sestavení webového serveru Apache 2.4.x

Jelikož operačním systémem pro Apache a Nginx webserver je Ubuntu, je možné použít baličkovací službu APT (Advanced Packaging Tool). Je nutné použít operační systém Ubuntu ve verzi 18.04 a novější. Server Apache lze nainstalovat pomocí

Výpis 5.1: Instalace serveru Apache 2

```
sudo apt-get install apache2
```

ve verzi 2.4.29, což neodpovídá nejaktuálnější verzi, která je 2.4.41. [38] Proto je nutné webové servery instalovat ručně. V následujících krocích je popsán postup instalace webového serveru a podrobná konfigurace se nachází v příloze ve složce `config/`.

1. Instalace základních knihoven, potřebných pro správné fungování webového serveru.
2. Instalace kihovny OpenSSL.
3. Instalace knihovny NGHTTP/2 jazyka C, která implementuje práci s HTTP/2 protokolem. [39]
4. Instalace balíku APR (Apache Portable Runtime Project APR). [40]
5. Instalace balíku APR-UTIL, což je knihovna obsahující mezilehlou vrstvu rozhraní pro operační systém. [40]
6. Instalace webového serveru Apache. [41]

5.3 Detekce dostupnosti webového serveru

Pro účely ověření dostupnosti webového serveru lze použít následující skript. Pokud webový server odpovídá na požadavky typu HTTP/2, skript vypíše „A“, pokud je nedostupný, vypíše „.“. Maximální doba, po kterou čeká na odpověď je nastavena na 0,5 sekundy.

Výpis 5.2: Detekce dostupnosti webového serveru

```
#!/bin/bash
while true; do if $(curl --http2-prior-knowledge \
--output /dev/null --silent --head \
--fail --max-time 0.5 http://10.0.0.2);
then printf "."; sleep 0.01;
else printf "A"; sleep 0.01;
fi; done;
```

Taktéž je možné ověřit veškerá otevřená TCP spojení se serverem pomocí programu `netstat`.

Výpis 5.3: Otevřené TCP spojení se serverem

```
$ netstat -natp | egrep ":80␣|␣ESTABLISHED" | \
> awk 'END{print␣NR}'
```

6 Generátor Slow DoS útoků

Náplní této části je návrh a řešení aplikace generátoru Slow DoS útoků na protokol HTTP/2 představených výše (kapitola 4). Aplikace je schopna útoky vygenerovat v množství, které si zvolí uživatel a při vyčerpání kapacity spojení, dojde k úspěšnému provedení útoku, nedostupnosti webového serveru po dobu uvedenou v tab. 4.1.

6.1 Funkcionalita aplikace

Aplikace je vytvořena v programovacím jazyce `python` ve verzi 3.7 s využitím volně dostupných knihoven v softwarovém repozitáři `PyPI` (Python Package Index). Algoritmus aplikace je uveden ve vývojovém diagramu C.1. Mezi základní funkcionality aplikace patří:

- zpracování vstupních parametrů,
- inicializace časovače, databáze a vytvoření instance útoku,
- průběh útoku,
- interpretace výsledků útoku a tvorba grafu účinku útoku.

6.1.1 Vstupní parametry

Zpracování vstupních parametrů uživatele je zajištěno knihovnou `argparse`, která umožňuje vývoj uživatelsky přívětivých konzolových programů. Ovládání aplikace je zajištěno pomocí pozičně závislých (positional) a volitelných (optional) parametrů.

Mezi pozičně závislé parametry patří:

- **HOST** – IP adresa oběti (serveru, na který chceme útočit).
- **ATTACK** – Typ Slow DoS útoku. Máme na výběr z parametrů `[read, post, preface, headers, settings]`, které odpovídají jednotlivým útokům.

Mezi volitelné parametry patří: 6.1

- `-h, --help` – Vytiskne nápovědu k programu.
- `-p, --port port` – Volitelný parametr, který specifikuje cílový port, na který útok bude probíhat. Výchozí hodnota je 80.
- `-o. --out file` – Volitelný parametr, který specifikuje název souborů, ve kterých bude uložena databáze (.csv) a vygenerovaný graf (.html). Výchozí hodnota je `output`.
- `-c, --connection connections` – Volitelný parametr udávající počet vytvořených paralelních útoků. Výchozí hodnota je 10.
- `-v, --verbal` – Parametr, který zajisté podrobný výpis činností programu do příkazové řádky.

Výpis 6.1: Obecné použití aplikace `slowhttp2test.py`

```
(venv) python3 slowhttp2test.py [-h] [-p port] [-o file] \
> [-c connections] [-v] HOST ATTACK
```

Ve výpise 6.2 je uveden příklad použití aplikace pro útok Slow PREFACE na server s ip adresou 10.0.0.2 a portem 80, 500 otevřenými spojeními a podrobným výpisem činností programu.

Výpis 6.2: Příklad provedení 500 Slow PREFACE útoků na webový server

```
(venv) python3 slowhttp2test.py 10.0.0.2 preface -p 80 \
> -o test -c 500 -v
```

6.1.2 Inicializace časovače, databáze, vytvoření instance útoku a jeho paralelní provedení

Hlavní program aplikace se nachází v souboru `slowhttp2test.py`. Tento program nejprve inicializuje časovač, databázi a následně vytvoří 2 paralelní procesy.

Třída `Timer()` vytváří časovač pro intervalové provádění útoků a požadavků na dostupnost webového serveru. Tento interval je tvořen tzv. časovými kroky. Výchozí hodnota těchto kroků je nastavena na 0,5 s. Databázová třída `CSVHandler()` má na starosti zapisování do databáze ve formátu `.csv`.

V souboru `slowhttp2attack.py` se nachází třída `Attack()`, jejíž vytvořená instance, s parametrem typu útoků, reprezentuje samotný Slow DoS útok na 1 TCP spojení. Aby útok byl úspěšný je nutné jej provést paralelně. K tomu použijeme knihovnu `pathos.multiprocessing`, která umožňuje vytváření paralelních procesů nad heterogenními zdroji dat, jako jsou například třídy. Další výhodou je, že umožňují sdílení globálních dat, např. řídících proměnných, napříč jednotlivými procesy pomocí metody `map()`, případně komunikaci mezi dvěma rozhraními pomocí metody `pipe()`, bez nutnosti synchronizace těchto procesů.

Hlavní proces č. 1 (`thread1()`) paralelně k procesu č. 2 (`thread2()`) vykonává měření času, dotazování se na dostupnost webového serveru a zápis do databáze. Proces č. 2 provádí vytváření paralelních spojení a následně vykonává útoky. Komunikace mezi hlavními procesy je řešena pomocí sdílených proměnných knihovny `multiprocessing`. Při každém zápisu do proměnné, dojde k uzamknutí daného procesu, abychom se vyvarovali problému zvanému MUTEX (MUTual EXclusion – vzájemné vyloučení procesů).

Dále je uveden příklad ukazující implementaci 1 TCP spojení s 1 HTTP/2 proudem, jehož následnou modifikací dostaneme útočící spojení obsloužené třídou `Attack()`.

Implementace 1 TCP spojení s 1 HTTP/2 proudem v jazyce Python 3

Základním prvkem následujícího programu, je síťový socket. Tento socket je koncovým bodem a využívá protokol TCP/IP. [1] Socket je určen cílovou IP adresou a číslem portu.

Jakmile je TCP spojení vytvořeno, byl vytvořen základní HTTP/2 proud. Nejprve inicializací objektu konfigurace proudu (`config`) a poté vytvořením instance samotného HTTP/2 proudu ze třídy `H2Connection()`. Tomuto objektu byly předány potřebné konfigurační parametry a vytvořen proud.

Jednotlivé proudy jsou na sobě nezávislé. Pokud je vytvořený proud hotový, je možné použít socket k odeslání těchto dat přes transportní vrstvu.

Nedefinované parametry spojení za nás vyplní podle RFC 7540 samotný klient Hyper-h2.[13] Dále se nachází ukázka vytvoření rámce záhlaví a možnost vytváření nových rámců či modifikací stávajících pomocí knihovny `hyperframe`. Všechny pozměněné rámce je nutno před samotným odesláním serializovat. U rámce záhlaví je nutno ještě použít binární kompresi HPACK. K tomu použijeme knihovnu `hpack` a třídy `Encoder()` a `Decoder()`.

Výpis 6.3: Příklad HTTP/2 komunikace

```
import socket
import h2.connection
from h2.config import H2Configuration
from hpack.hpack_compat import Encoder
from hyperframe.frame import HeadersFrame

ADDRESS = '10.0.0.2'
PORT = 80

def main():
with socket.create_connection((ADRESS, PORT)) as s:
config = H2Configuration()
http2_connection = h2.connection. \
H2Connection(config=config)
http2_connection.initiate_connection()
# Zde se nachází prostor pro přídávání,
# či upravování rámců
s.sendall(http2_connection.data_to_send())

# Paralelní http2 stream s ID o 1 větší
```

[1] Pro samotný přenos potřebuje znát IP adresu cílové stanice a tedy využije také síťové vrstvy.

```
headers = [
(":authority",ADDRESS),
(":path","/"),
(":scheme","http"),
(":method","GET")
]
# Při úpravě hlavičky je nutnou použít binární
# kodér HPACK
e = Encoder()
hf = HeadersFrame(1)
hf.data = e.encode(headers)
http2_connection._data_to_send += hf.serialize()
s.sendall(http2_connection.data_to_send())

if __name__ == '__main__':
main()
```

6.1.3 Interpretace výsledků a tvorba grafu

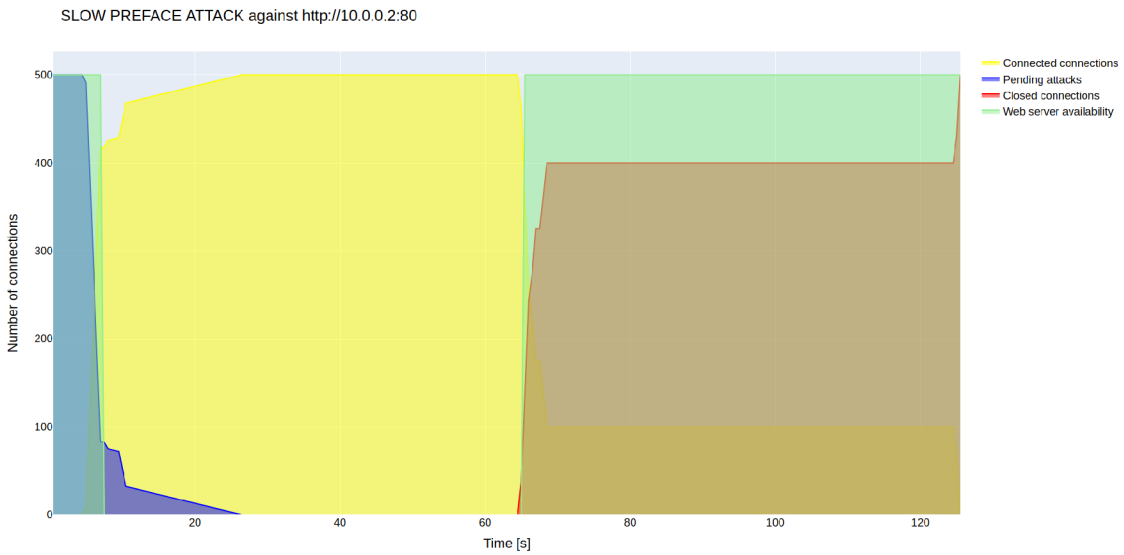
Naměřená data jsou postupně ukládána do databáze (.csv soubor). Tento soubor obsahuje naměřená data hlavním programem `slowhttp2test.py`. Jednotlivé sloupce databáze:

- `Time` – časový údaj, generovaný časovačem,
- `Connected` – počet úspěšně vytvořených a připojených spojení k cílovému serveru (odpovídá čáře `Connected connections`),
- `Pending` – počet čekajících spojení (útoků) (odpovídá čáře `Pending attacks`),
- `Closed` – počet zavřených spojení (odpovídá čáře `Closed connections`),
- `Service_Available` – udává dostupnost webového serveru (opdovídá čáře `Web server availability`).

Výsledný graf je potom závislost útoků (typů spojení) na čase.6.1 Dostaneme jej například s nastavením programu `slowhttp2test.py` ve výpisu 6.4

Výpis 6.4: Parametry skriptu `slowhttp2test.py` Slow DoS útoku typu Slow PREFACE s 500 otevřenými spojeními.

```
python3 slowhttp2test.py 10.0.0.2 preface -p 80 -o test -c 500
```



Obr. 6.1: Graf útoku Slow PREFACE

Z grafu je patrné, že útok Slow PREFACE probíhal po dobu 60 sekund. Tato hodnota odpovídá naměřené hodnotě z tabulky 4.1. V levé části lze pozorovat postupné nabíhání jednotlivých útoků. V pravé potom postupné zavírání otevřených spojení a zároveň dostupnost webového serveru.

Proč nedochází k uzavření všech spojení ihned po dokončení útoku? Pro uzavírání jednotlivých TCP spojení je použita metoda `socket.close(tcp_conn)`, která plně neuzavře toto spojení. Třída `Manager()` udržuje referenci na část spojení. Abychom docílili uzavření spojení ihned potom, co server obnoví spojení, museli bychom nahradit třídu `multiprocessing.Process()` třídou `multiprocessing.Queue()`.

6.2 Použité knihovny

Jelikož jsou útoky implementovány v jazyce `python` ve verzi 3.7, byly použité, kromě standardních knihoven, i veřejně dostupné knihovny pro práci s HTTP/2 protokolem. Ukázka implementace jednotlivého TCP spojení s jedním HTTP/2 proudem následuje ve výpisu 6.3.

- **Hyper-h2** – Knihovna `hyper-h2` je univerzální knihovna pro práci s protokolem HTTP/2. Je nezávislá na implementaci síťové vrstvy, která parsuje data nebo na vícevláknovém modelu. [42]
- **Hyperframe** – Hyperframe je nástroj pro práci s HTTP/2 rámci. Umožňje vytvářet, upravovat a měnit hodnoty určitého příznaku, serializaci a parsování rámců. Stejně jako `hyper-h2` se jedná o čistou implementaci v jazyce `python`, tudíž je nezávislý na počtu spojení.[43]

- **Hpack** – Knihovna `hpack` poskytuje jednoduché rozhraní kompresnímu algoritmu HPACK, který je použit pro kompresi HTTP hlaviček. Mezi dvě základní třídy patří `Decoder()` a `Encoder()`.[44]
- **Pandas** – Knihovna `pandas` slouží k práci se strukturovanými daty. Umožňuje práci s datovými soubory, jejich konverzí do strojově dobře reprezentovatelného formátu. [49]
- **Plotly** – `Plotly` je interaktivní grafická knihovna postavena na JavaScriptu. Umožňuje generování a tisk 30 druhů grafů, obrázků a schémat. [50]
- **Pathos** – Knihovna `pathos` poskytuje vysokoúrovňové rozhraní pro práci s různorodými paralelními procesy. [51]

7 Detekce Slow DoS útoků

Slow DoS útoky většinou cílí na chybnou implementaci internetového protokolu, nové typy dosud neobjevených zranitelností, zneužití principů vytváření spojení, či samotné vytížení co největšího počtu otevřených komunikačních kanálů. Těchto útoků a jejich variací může existovat velké množství. Tudíž útočníkům naskýtá možnost efektivně útočit.

Jelikož tyto útoky generují malý síťový provoz a jsou tím pádem téměř nerozeznatelné od běžného provozu, je velice obtížné tyto útoky detekovat.

7.1 Obrana proti Slow DoS útokům

Obrana proti DoS útokům by měla být schopna v ideálním případě detekovat všechny typy útoků. Dále by měla být schnopna efektivně rozlišit mezi validním a nevalidním provozem, a tento provoz odfiltrovat, tak aby nedošlo k ovlivnění validních požadavků a zároveň žádný nevalidní provoz nepronikl skrz obranu. Mechanizmy obrany by neměly příliš čerpat zdroje oběti. Tedy oběť, na které běží účinný obranný mechanismus, by neměla pociťovat jeho činnost.

Základní a nejjednodušší obranou proti Slow DoS útokům je udržovat daný počítačový systém aktuální. Zároveň je to výzvou pro tvůrce webových serverů aby reagovali co možná nejpromtněji na zveřejněné bezpečnostní hrozby formou bezpečnostních aktualizací.

Další možností je využití bezpečnostních modulů a systémů třetích stran. Mezi nejjednodušší patří systémy založené na detekci otevřených spojení z určitých IP adres a následná filtrace těchto zařízení. Tento přístup nebývá moc efektivní. Umožňuje pouze zmenšit rozsah útoku. Další metodou je využití systémů detekce/prevence průniku (IDS - Intrusion Detection System / IPS - Intrusion Prevention System).

Dále existuje několik zásad, jak zabezpečit webový server. [46]

- Omezit dobu otevřených spojení s určitou IP adresou na nezbytně nutnou. Např. pomocí nástroje `iptables`.
- Omezit počet otevřených spojení z určité IP adresy.
- Omezit maximální počet otevřených spojení na jednoho uživatele.
- Omezit dobu čekání serveru na příjem/odeslání požadavku. Po vypršení této doby, spojení uzavírat.
- Využít maximální výpočetní kapacitu serveru pro obsloužení co největšího počtu uživatelů.
- Vytvořit tzv. `HONEYPOT` servery uvnitř sítě.

7.2 Metody detekce Slow DoS útoků

Jelikož nejvíce zranitelným webovým serverem je Apache ve verzích 2.4.17 a 2.4.18 následující detekce Slow Dos útoků je primárně určená právě pro tyto servery.

Základní metody detekce můžeme rozdělit na přímé a nepřímé.

Mezi přímé metody řadíme takové, které ať už v reálném čase nebo zpětně ze zachyceného síťového provozu, detekují jednotlivé útoky. Jelikož Slow DoS útoky jsou nejvýznamnější na úrovni aplikační vrstvy, protokolu HTTP/2, budeme provádět detekci významných parametrů každého z útoků.

Mezi nepřímé metody řadíme zpravidla metody založené na statistickém zpracování dat, případně metody strojového učení, kdy na základě množství a velikosti paketů z určité IP adresy můžeme rozhodnout, s určitou pravděpodobností, zda-li se jedná o podezřelé chování, či nikoli.

Způsobů, jak můžeme tyto útoky detekovat je víc. Lze například vytvořit modul serveru Apache, který bude mít na starosti tuto detekci a následnou filtraci provozu. Automatizovaný skript, který bude tyto útoky detekovat v reálném čase nebo zpětně ze souboru zachycených paketů. Také je možné využít prvek aplikační proxy, který bude síťový provoz filtrovat ještě dříve, než dorazí k webovému serveru.

7.3 Nástroje pro zachytávání a analýzu paketů

Základním nástrojem pro analýzu paketů je grafický program Wireshark. Základem zachytávání paketů je knihovna `libpcap` pro linuxové operační systémy a `winpcap` pro operační systém Windows. [52] Tuto knihovnu obsahuje program `tcpdump`, který funguje jako analyzátor síťové komunikace pomocí příkazové řádky. Obdoba programu `tcpdump` je program `TShark`, který umožňuje analýzu paketů v reálném čase i ze souborů obsahujicích zachycené pakety. [53] Formát, který `TShark` používá je `.pcapng`. Jedná se o stejný formát, který používá Wireshark.

Nástroj, který umožňuje zachytávat pakety a následně je zpracovávat je program `pyshark`, který pro práci s pakety využívá program `TShark`. Podporuje analýzu paketů v reálném čase i ze souborů. Výchozím formátem pro zachytávané pakety je formát `.pcap`. [54]

7.4 Detektor Slow DoS útoků

Pro detekci Slow DoS útoků je použit programovací jazyk `python` ve verzi 3.7 s využitím volně dostupných knihoven. Základní knihovnou detektoru je `pyshark`.

Pyshark nám nabízí detekci paketů v reálném čase a nebo ze souboru. Pro detekci v reálném čase musíme specifikovat rozhraní, na kterém bude docházet k zachytávání paketů. Dále můžeme specifikovat aplikovatelný filtr, stejný jako v programu Wireshark, pro filtraci příchozího a odchozího provozu. Jelikož výše představené Slow DoS útoky jsou generovány čistě ze strany útočníka, omezíme se pouze na analýzu příchozích paketů obsahujících pole HTTP/2 aplikační vrstvy a cílovou adresu (destination) serveru jako aplikovatelný filtr.

Výpis 7.1: Inicializace a zachytávání paketů na rozhraní `ens33` s aplikovaným filtrem po dobu 20 sekund.

```
capture = pyshark.LiveCapture(interface='ens33', \
display_filter='ip.dst␣==␣10.0.0.2␣&&␣http2')
capture.sniff(timeout=20)
```

Pro detekci ze souboru musíme specifikovat soubor ve formátu `.pcap`. Můžeme taktéž použít aplikovatelný filtr.

Výpis 7.2: Detekce HTTP/2 paketů s aplikovaným filtrem.

```
capture = pyshark.FileCapture('capture_file.pcap', \
display_filter='ip.dst␣==␣10.0.0.2␣&&␣http2')
```

7.4.1 Vybrané vlastnosti Slow DoS útoků

Pro každý z výše představených útoků je nutné zvolit parametr rámců, případně samotné rámce sloužící pro detekci útoků. [48]

a) Útok Slow READ – parametr `SETTINGS_INITIAL_WINDOW_SIZE: 0` v HTTP/2 rámci `SETTINGS`.

Výpis 7.3: Detekce parametru výchozí velikosti okna.

```
packet['http2'].settings_initial_window_size == '0'
```

b) Útok Slow POST – parametr `END_STREAM: 0` a `END_HEADERS: 1` v HTTP/2 rámci `HEADERS`.

Výpis 7.4: Detekce parametrů ukončení proudu a rámce hlaviček.

```
packet['http2'].flags_end_stream == '0'
packet['http2'].flags_eh == '1'
# NEBO
packet['http2'].flags == '0x00000004'
```

c) Útok Slow PREFACE – pouze samostatný výskyt rámce `CONNECTION_PREFACE`. Pro jednoznačnou detekci se nesmí na stejném HTTP/2 paketu vyskytovat

žádný další rámec. Tzn. budeme testovat existenci rámce `SETTINGS`, jehož nepřítomnost by měla vyvolat chybu `ATTRIBUTE ERROR`.

Výpis 7.5: Detekce samostatného rámce pro inicializaci HTTP/2 spojení.

```
packet['http2'].magic == \
'PRI␣*␣HTTP/2.0\\xd\\xa\\xd\\xaSM\\xd\\xa\\xd\\xa'
packet['http2'].settings # Attribute Error
```

d) Útok Slow HEADERS – parametr `END_STREAM: 1` a `END_HEADERS: 0` v HTTP/2 rámci `HEADERS`.

```
packet['http2'].flags_end_stream == '1'
packet['http2'].flags_eh == '0'
# NEBO
packet['http2'].flags == '0x00000001'
```

e) Útok Slow SETTINGS – nepotvrzený rámec `SETTINGS`. Jelikož jsme se zaměřili pouze na detekci spojení směrem k serveru, musíme kontrolovat IP adresu, hodnotu portu a číslo streamu odchozího spojení. Útočník provede zahájení spojení pomocí rámce `MAGIC` a `SETTINGS`. Server vyšle svůj a potvrzovací rámec `SETTINGS`. Útočník jej nepotvrdí. Tzn. na stejném spojení nelze detekovat další zahajovací rámec HTTP/2 spojení ani žádný jiný `SETTINGS` rámec – `Attribute Error`.

```
# 1. paket
packet['http2'].magic == \
'PRI␣*␣HTTP/2.0\\xd\\xa\\xd\\xaSM\\xd\\xa\\xd\\xa'
packet['http2'] == settings_initial_window_size != '0'
packet['ip'].src == source_address
packet['tcp'].port == str(port_number)
packet['http2'].streamid == str(stream_number)

# n. paket
packetn['ip'].src == source_address
packetn['tcp'].port == str(port_number)
packetn['http2'].streamid == str(stream_number)
packetn['http2'].settings # Attrbute Error
```

7.4.2 Funkcionalita aplikace

Aplikace pro detekci Slow DoS útoků na protokol HTTP/2 je vytvořena v programovacím jazyce `python` ve verzi 3.7 s využitím volně dostupných knihoven v softwaro-

vém repozitáři `PyPI`. Algoritmus detekce jednotlivých útoků je uveden ve vývojovém diagramu C.2. Mezi základní funkcionality aplikace patří:

- zpracování vstupních parametrů,
- zvolení způsobu detekce (online/offline),
- analýza kolekce zachycených paketů a vytvoření statistiky jednotlivých útoků,
- tvorba grafu závislosti počtu validních HTTP/2 požadavků a modifikovaných HTTP/2 požadavků.

Vstupní parametry

Zpracování vstupních parametrů uživatele je zajištěno knihovnou `argparse`, která umožňuje vývoj uživatelsky přívětivých konzolových programů. Parametry dělíme podle toho, zda-li volíme online detekci Slow DoS útoků nebo offline ze souboru.

Pro online detekci jsou následující parametry.

- **`-l, --live`** – Parametr určující činnost detektoru v reálném čase.
- **`-i, --interface`** `interface` – Parametr, jehož hodnota specifikuje název rozhraní, na kterém bude probíhat detekce útoků.
- **`-d, --duration`** `duration` – Parametr, jehož hodnota určuje dobu detekce útoků. Výchozí hodnota je nastavena na 60 sekund.

Pro offline detekci jsou následující parametry.

- **`-f, --file`** `file_name` – Parametr offline detekce útoků, jehož hodnota určuje název souboru obsahující zachytávaná síťová data.
- **`-a, --address`** `ip_address` – Parametr, který určuje IP adresu serveru, u něhož má být provedená detekce útoků.

Interpreteace výsledků a tvorba grafu

Data, ať už z odchytávání paketů v reálném čase, nebo ze souboru, jsou detektorem analyzována. Výstupem je slovník obsahující údaje o zachycených útocích, ze kterého je následně generován graf.

Výpis 7.6: Slovník obsahující detekci útoků.

```
captured_attacks = {
  "SLOW_READ": 0,
  "SLOW_POST": 0,
  "SLOW_PREFACE": 0,
  "SLOW_HEADERS": 0,
  "SLOW_SETTINGS": 0,
  "NORMAL_COMMUNICATION": 0
}
```

Provedeme detekci Slow DoS útoků vygenerovaných pomocí skriptu `slowhttp2test.py` s následujícími náhodně zvolenými hodnotami parametrů.

Výpis 7.7: Vytvoření Slow DoS útoků s náhodně zvolenými hodnotami parametrů.

```
python3 slowhttp2test.py 10.0.0.2 read -c 12
python3 slowhttp2test.py 10.0.0.2 post -c 18
python3 slowhttp2test.py 10.0.0.2 preface -c 8
python3 slowhttp2test.py 10.0.0.2 headers -c 14
python3 slowhttp2test.py 10.0.0.2 settings -c 10
```

A s validními HTTP/2 požadavky vygenerovanými softwarovým nástrojem `cURL`.

Výpis 7.8: Generování validních HTTP/2 požadavků v intervalu 1 sekundy.

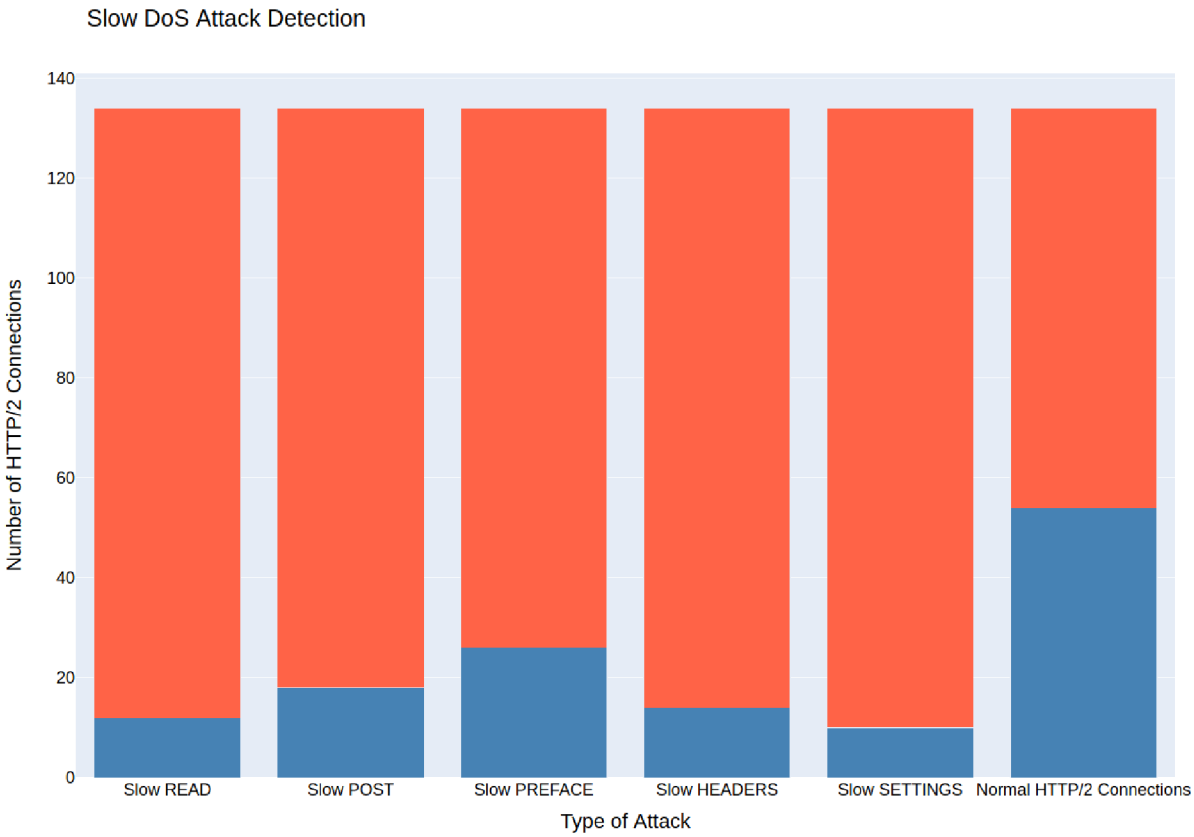
```
$ while true; do curl --http2 http://10.0.0.2; sleep 1; done
```

Detekci útoků provedeme pomocí programu `slowhttp2detect.py` s následujícím nastavením.

Výpis 7.9: Detekce Slow DoS útoků na rozhraní `ens33` po dobu 20 sekund.

```
python3 slowhttp2detect.py -l -i ens33 -d 20
```

Výsledkem je graf online detekce Slow DoS útoků 7.1.



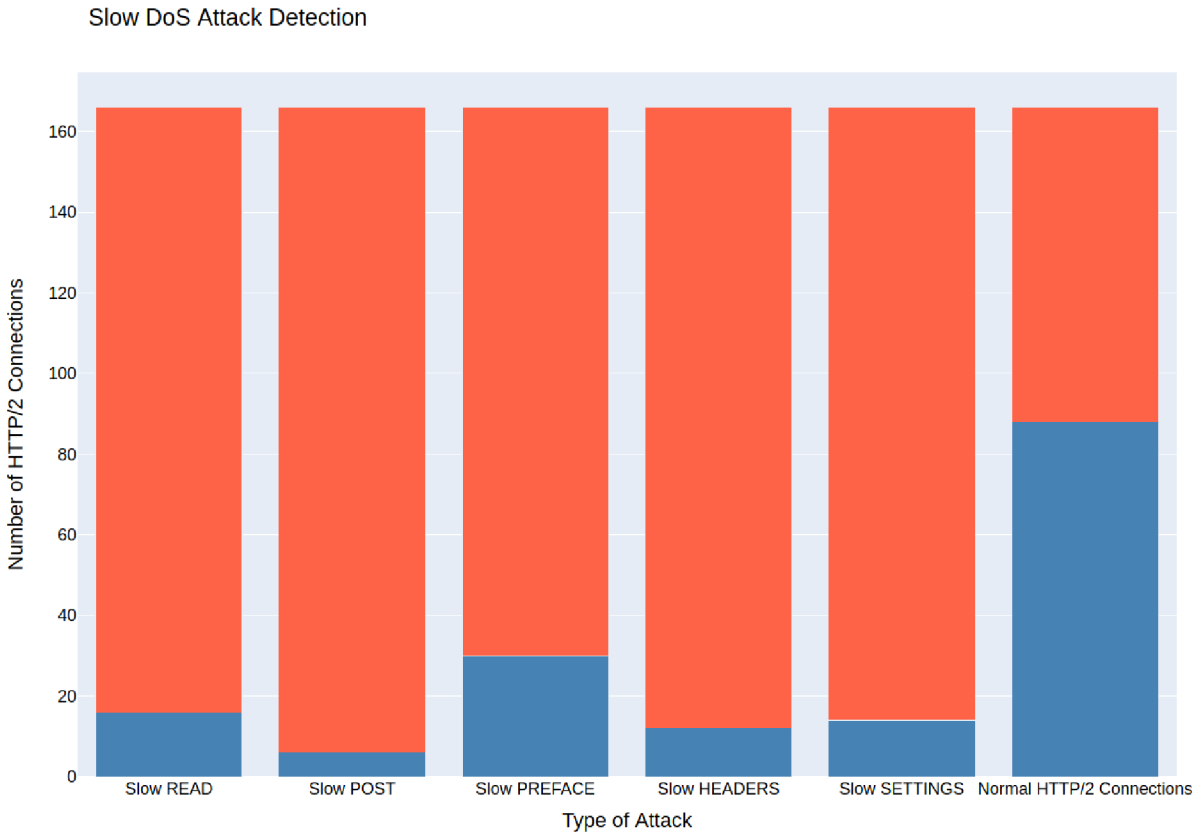
Obr. 7.1: Graf online detekce Slow DoS útoků

Obdobně vyzkoušíme offline detekci. Nejprve si vygenerujeme Slow DoS útoky spolu s validním HTTP/2 spojením.

Výpis 7.10: Vytvoření Slow DoS útoků s náhodně zvolenými parametry.

```
python3 slowhttp2test.py 10.0.0.2 read -c 16
python3 slowhttp2test.py 10.0.0.2 post -c 6
python3 slowhttp2test.py 10.0.0.2 preface -c 8
python3 slowhttp2test.py 10.0.0.2 headers -c 12
python3 slowhttp2test.py 10.0.0.2 settings -c 14
```

Následně obdobným způsobem vygenerujeme validní HTTP/2 provoz. Výsledkem je graf offline detekce Slow DoS útoků 7.2.



Obr. 7.2: Graf offline detekce Slow DoS útoků

8 Závěr

Cílem bakalářské práce bylo prostudovat principy Slow DoS útoků, popsat rozdíly mezi klasickým DoS a DDoS útokem. Dále se zaměřit na implementaci pěti představených útoků a otestovat úspěšnost útoků na různé typy webových serverů. Navrhnout a vytvořit funkční generátor útoků a následně sestavit metody jejich detekce spolu s návrhem a implementací detektor těchto útoků.

V úvodu práce byla provedena stručná charakteristika síťového provozu. Je zde popsán rozdíl mezi modelem ISO/OSI a TCP/IP. Dále je uveden princip navázání spojení na úrovni transportní vrstvy.

Další kapitola se zabývá rozdělením komunikace na aplikační vrstvě pomocí hypertextového protokolu a jsou zde popsány jednotlivé rozdíly. Nejdůležitějším protokolem pro tuto práci je HTTP/2. Dále je vysvětlen základní princip komunikace pomocí protokolu HTTP/2.

Další kapitola popisuje rozdělení útoků na DoS a DDoS útoky z pohledu jednotlivých vrstev a rozdělení Slow DoS útoků na jednotlivé typy.

V následující kapitole jsou popsány Slow DoS útoky na protokol HTTP/2. Tato práce se konkrétně zabývá útoky Slow READ, Slow POST, Slow PREFACE, Slow HEADERS a Slow SETTINGS. Dále bylo provedeno srovnání jednotlivých útoků z pohledu odezvy webových serverů na jednotlivý útok. A byl popsán účinek jednotlivých útoků na webový server.

V další kapitole bylo popsáno testovací prostředí - lokální virtualizovaná síť. Dále je zde uveden postup sestavení webového serveru ze zdrojových souborů, metoda detekce dostupnosti webového serveru.

Další kapitola představuje návrh a implementaci generatoru Slow DoS útoků. Popisuje návrh aplikace, způsob spuštění a reprezentaci výsledků a tvorbu grafu.

Poslední kapitola se zabývá návrhem a implementaci detektoru Slow DoS útoků. Zahrnuje popis vlastností detektoru, způsob spuštění aplikace a na závěr interpretuje výsledky detekce.

Závěrem bylo dosaženo úspěšného vygenerování Slow DoS útoků proti webovému serveru Apache 2.4.17, odepření služeb tohoto serveru a následně byla provedena detekce a klasifikace jednotlivých druhů útoků.

Předmětem dalšího vývoje může být vytvoření aktivní obrany v reálném čase, zdokonalení detekce statistickými metodami nebo strojovým učením, zahrnout do detekce všechny varianty síťového provozu s využitím zabezpečeného spojení HTTPS. Další možností je přidat do detektoru další typy Slow DoS útoků představené společností Netflix v srpnu 2019. [45]

Literatura

[1] CONRAD, Eric, Seth MISENAR a Joshua FELDMAN.: *CISSP study guide*. 2nd ed. Waltham, Mass: Syngress, imprint of Elsevier, 2012. ISBN 978-1-59749-961-3.

[2] *TCP Connection Establish and Terminate.* [online]. [cit. 1. 11. 2019]. Dostupné z URL: <https://www.vskills.in/certification/tutorial/information-technology/basic-network-support-professional/tcp-connection-establish-and-terminate/>

[3] *An overview of HTTP*, Mozilla Corporation, MDN, [online]. Zveřejněno 29. 9. 2019 [cit. 1. 11. 2019]. <https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview>

[4] Engström N.: *What is difference between URI, URL, URN ?*, Stack Overflow, [online]. Zveřejněno 22. 7. 2017 [cit. 2. 11. 2019]. Dostupné z URL: <https://stackoverflow.com/questions/4913343/what-is-the-difference-between-uri-url-and-urn>

[5] Berners-Lee T, et al.: *Hypertext Transfer Protocol – HTTP/1.0*, IETF, 60 stran, [online]. Zveřejněno 5. 1996 [cit. 2. 11. 2019]. Dostupné z URL: <https://tools.ietf.org/html/rfc1945>

[6] Berners-Lee T, et al.: *Hypertext Transfer Protocol - HTTP/1.1*, IETF, 176 stran, [online]. Zveřejněno 6. 2019 [cit. 3. 11. 2019]. Dostupné z URL: <https://tools.ietf.org/html/rfc2616>

[7] *Connection management in HTTP/1.x*, Mozilla Corporation, MDN, [online]. Zveřejněno 27. 11. 2019 [cit. 3. 11. 2019]. Dostupné z URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Connection_management_in_HTTP_1.x>

[8] *Usage of site elements for websites*, W3Techs, [online]. Zveřejněno 10. 12. 2019 [cit. 10. 12. 2019]. Dostupné z URL: <https://w3techs.com/technologies/overview/site_element>

[9] Gamage T. A.: *Evolution of HTTP - HTTP/0.9, HTTP/1.0, HTTP/1.1*, [online]. Zveřejněno 17. 11. 2017 [cit. 4. 11. 2019]. Dostupné z URL: <https://medium.com/platform-engineer/evolution-of-http-69cfe6531ba0>

[10] Langley A, Riddoch A, Wilk A., Krasic Ch., Zhang D.: *The QUIC Transport Protocol: Design and Internet-Scale Deployment*,

Google, [online]. [cit. 14. 12. 2019]. Dostupné z URL: <https://storage.googleapis.com/pub-tools-public-publication-data/pdf/8b935debf13bd176a08326738f5f88ad115a071e.pdf>

[11] *SPDY: An experimental protocol for a faster web* [online]. [cit. 13. 12. 2019]. Dostupné z URL: <https://www.chromium.org/spdy/spdy-whitepaper>

[12] Moučka M.: *Laboratorní scénáře umožňující srovnání protokolů přenosu webových stránek*: diplomová práce. Brno, Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2016, 88 stránek, [online] [cit. 16. 12. 2019]. Dostupné z URL: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=147442>

[13] RFC 7540 *Hypertext Transfer Protocol Version 2 (HTTP/2)* IETF, 96 stran, [online]. Zveřejněno 5. 2015 [cit. 2. 11. 2019]. Dostupné z URL: <https://tools.ietf.org/html/rfc7540

[14] Theedom A: *HTTP/2 under the hood* [online]. Zveřejněno 29. 6. 2017 [cit. 14. 12. 2019]. Dostupné z URL: <https://developer.ibm.com/articles/wa-http2-under-the-hood/>

[15] POLLARD B.: *HTTP/2 in Action.* Shelter Island, NY: Manning Publications Co., [2019]. 416 s. ISBN 9781617295164.

[16] Stenberg D.: *HTTP/2 Explained.* [online] Zveřejněno 3. 3. 2019 [cit. 25. 11. 2019]. Dostupné z URL: <https://http2-explained.haxx.se/content/en/

[17] Jirsák F.: *Jak funguje nový protokol HTTP/2* [online]. Zveřejněno 4. 3. 2015 [cit. 13. 12. 2019]. Dostupné z URL: <https://www.root.cz/clanky/jak-funguje-novy-protokol-http-2/>

[18] Grigorik I., Surma: *Introduction to HTTP/2*, [online]. [cit. 3. 12. 2019]. Dostupné z URL: <https://developers.google.com/web/fundamentals/performance/http2>

[19] Krasnov V.: *HPACK: the sillent killer (feature) of HTTP/2*, Cloudlfare Blog, [online]. Zveřejněno 28. 11. 2016 [cit. 5. 12. 2019]. Dostupné z URL: <https://blog.cloudflare.com/hpack-the-silent-killer-feature-of-http-2/>

[20] *What is QUIC?* [online]. [cit. 14. 12. 2019]. Dostupné z URL: <https://docs.google.com/document/d/1gY9-YNDNAB1eip-RTPbqphgySwSNSDHLq9D5Bty4FSU/edit>

[21] Stenberg D.: *HTTP/3 Explained.* [online] Zveřejněno 25. 11. 2019 [cit. 27. 11. 2019]. Dostupné z URL: <https://http3-explained.haxx.se/en/

[22] Ghendini A.: *HTTP/3: the past, the present, and the future*, Cloudflare Blog, [online]. Zveřejněno 26. 9. 2019 [cit. 4. 12. 2019]. Dostupné z URL: <https://blog.cloudflare.com/http3-the-past-present-and-future/>

[23] *What is a Denial-of-Service (DoS) Attack?* [online]. [cit. 15. 12. 2019] Dostupné z URL: <https://www.cloudflare.com/learning/ddos/glossary/denial-of-service/>

[24] Dobeš J.: *Analýza nástrojů provádějících pomalé DoS útoky* [online] Zveřejněné 1. 6. 2015 [cit. 15. 12. 2019] Dostupné z URL: <https://is.muni.cz/th/ptl75/DP_text_sajtkgzq.pdf>

[25] TRIPATHI N., HUBBALLI N.: *Slow rate denial of service attacks against HTTP/2 and detection.* Computer & Security, vol.: 72, 2018, 255-272, [online]. Zveřejněno 28. 9. 2017 [cit. 1. 11. 2019]. Dostupné z URL: <https://doi.org/10.1016/j.cose.2017.09.009>

[26] IMPERVA Hacker Intelligence Initiative: *HTTP/2: In-depth analysis of the top four flaws of the next generation web protocol*, [online]. [cit. 15. 11. 2019]. Dostupné z URL: <https://www.imperva.com/docs/Imperva_HII_HTTP2.pdf>

[27] Winkel S.: *Network Forensics and HTTP/2* SANS Institute, [online]. Zveřejněno 27. 12. 2017. [cit. 17. 11. 2019] Dostupné z URL: <https://www.sans.org/reading-room/whitepapers/forensics/network-forensics-http-2-36647>

[28] National Vulnerability Database *CVE-2016-1546* NIST, [online]. Zveřejněno 6. 7. 2019 [cit. 4. 12. 2019]. Dostupné z URL: <https://nvd.nist.gov/vuln/detail/CVE-2016-1546>

[29] Softvelum: *Implementace HTTP/2 protokolu.* [online] Zveřejněno 21. 11. 2019 [cit. 23. 11. 2019]. Dostupné z URL: <https://github.com/http2/http2-spec/wiki/Implementations>

[30] *apache2.conf*, [online]. [cit. 3. 12. 2019]. Dostupné z URL: <https://www.linode.com/docs/assets/apache2.conf>

[31] *Multi-Processing Modules (MPMs)*, Apache HTTP Server Project, [online]. [cit. 5. 12. 2019]. Dostupné z URL: <https://httpd.apache.org/docs/2.4/mpm.html>

[32] *Apache MPM Common Directives*, Apache HTTP Server Project, [online]. [cit. 5. 12. 2019]. Dostupné z URL: <http://httpd.apache.org/docs/current/mod/mpm_common.html#maxclients>

[33] Geerling J., *3 Small Tweaks to make Apache fly*, [online]. Zveřejněno 5. 4. 2013 [cit. 5. 12. 2019]. Dostupné z URL: <https://www.jeffgeerling.com/blog/3-small-tweaks-make-apache-fly>

[34] *Použití protokolu HTTP/2 jako prvek webové technologie*, [online]. Zveřejněno 8. 12. 2019 [cit. 8. 12. 2019]. Dostupné z URL: <https://w3techs.com/technologies/details/ce-http2>

[35] *Druhy webových serverů v produkčním prostředí*, [online]. Zveřejněno 8. 12. 2019 [cit. 8. 12. 2019]. Dostupné z URL: <https://w3techs.com/technologies/overview/web_server>

[36] *Použití operačního systému*, W3Techs, [online]. Zveřejněno 8. 12. 2019 [cit. 8. 12. 2019]. Dostupné z URL: <https://w3techs.com/technologies/overview/operating_system>

[37] *Podverze webového serveru Apache 2.4*, [online]. Zveřejněno 2. 12. 2019 [cit. 2. 12. 2019]. Dostupné z URL: <https://w3techs.com/technologies/details/ws-apache/2.4>

[38] *Apache httpd 2.4.41 Released*, Apache HTTP Server Project, [online]. [cit. 6. 12. 2019]. Dostupné z URL: <https://httpd.apache.org/>

[39] *nghttp2 - HTTP/2 C Library*, [online]. [cit. 7. 12. 2019]. Dostupné z URL: <https://github.com/nghttp2/nghttp2>

[40] *Apache APR Project Source Code Distributions*, Apache HTTP Server Project, [online]. [cit. 7. 12. 2019]. Dostupné z URL: <https://archive.apache.org/dist/apr/>

[41] *Apache HTTP Server Source Code Distributions*, Apache HTTP Server Project, [online]. [cit. 7. 12. 2019]. Dostupné z URL: <https://archive.apache.org/dist/httpd/>

[42] *hyper-h2: A pure-Python HTTP/2 protocol stack.* [online]. [cit. 8. 12. 2019]. Dostupné z URL: <https://python-hyper.org/projects/h2/en/stable/>

[43] *hyperframe: HTTP/2 Framing for Python*, [online]. [cit. 8. 12. 2019]. Dostupné z URL: <https://python-hyper.org/projects/hyperframe/en/latest/>

[44] *hpack: HTTP/2 Header Compression for Python*, [online]. [cit. 8. 12. 2019]. Dostupné z URL: <https://python-hyper.org/projects/hpack/en/latest/>

[45] *HTTP/2 Denial of Service Advisory.* [online]. Zveřejněno 3. 8. 2019 [cit. 8. 12. 2019]. Dostupné z URL: <https://github.com/Netflix/security-bulletins/blob/master/advisories/third-party/2019-002.md>

[46] *A Review of Defense Against Slow HTTP Attack.* International Journal on Informatics Visualization, vol. 1, 2017, 127, [online]. Zveřejněno 11. 2017 [cit. 2. 5. 2020]. Dostupné z URL: <https://joiv.org/index.php/joiv/article/view/51>

[47] Potter J., *Apache Performance Tuning*, series I. – V., [online]. Zveřejněno 18. 9. 2018 [cit. 10. 5. 2020]. Dostupné z URL: <https://www.liquidweb.com/kb/apache-performance-tuning-swap-memory/>

[48] Kumar Ch., *Artificial Intelligence: Definition, Types, Examples, Technologies*, Medium, [online]. Zveřejněno 31. 8. 2018 [cit. 12. 5. 2020]. Dostupné z URL: <https://medium.com/@chethankumargn/artificial-intelligence-definition-\types-examples-technologies-962ea75c7b9b>

[49] *pandas 1.0.3 - Project description*, [online]. [cit. 22. 5. 2020]. Dostupné z URL: <https://pypi.org/project/pandas/>

[50] *plotly 4.7.1 - Project description*, [online]. [cit. 22. 5. 2020]. Dostupné z URL: <https://pypi.org/project/plotly/>

[51] *pathos 0.2.5 - Project description*, [online]. [cit. 22. 5. 2020]. Dostupné z URL: <https://pypi.org/project/pathos/>

[52] *TCPDUMP & LIBPCAP*, [online]. [cit. 23. 5. 2020]. Dostupné z URL: <https://www.tcpdump.org/>

[53] *tshark - Dump and analyze network traffic*, [online]. [cit. 23. 5. 2020]. Dostupné z URL: <https://www.wireshark.org/docs/man-pages/tshark.html>

[54] Green D., *pyshark - Python wrapper for tshark*, [online]. Zveřejněno 15. 5. 2020 [cit. 23. 5. 2020]. Dostupné z URL: <https://github.com/KimiNewt/pyshark/commits/master>

Seznam symbolů, veličin a zkratek

HTTP/2 Hyper Text Transport Protocol 2. generace je novým protokolem pro přenos webových stránek mezi klientem a serverem.

RFC Request For Comments je označení dokumentů popisujících internetové protokoly. Vydává je organizace IETF a jsou považovány za standardy.

Slow DoS Často velmi dobře skryté útoky cílené na aplikační vrstvu s relativně malým provozem.

ISO/OSI Jedná se o referenčí model síťové komunikace v počítačových a telekomunikačních sítích.

TCP Transmission Control Protocol je protokolem transportní vrstvy modelu ISO/OSI. Tento protokol garantuje spolehlivé doručení paketů.

HTML Hypertext Markup Language je typ značkovacího jazyka používaného pro tvorbu webových stránek. Základní vlastností strukturovanost a propojenost dokumentů napsaných pomocí jazyka HTML.

WWW World Wide Web je celosvětová síť pro prohlížení, ukládání a odkazování dokumentů v internetu.

URL Uniform Resource Locator je řetězec znaků, který slouží k identifikaci zdroje informací na internetu.

URI Uniform Resource Identifier je řetězec znaků určený ke specifikování zdroje informací v rámci internetu.

FIFO First In, First Out je princip zařazování datových bloků do fronty. Ten, který vejde první do fronty tuto frontu jako první opustí.

IPS Intrusion Prevention System je systém pro detekci a prevenci průniku. Známý také jako Intrusion Detection System (IDS) je systém pro detekci škodlivého síťového provozu. Funguje na principu inspekce síťového provozu a případné filtrace paketů, opravy kontrolního součtu, přeřazení paketů, resetování spojení, či zastavení komunikace.

HTTPS Hypertext Transfer Protocol Secure je protokol umožňující šifrovanou HTTP komunikaci. Využívá protokolu SSL (Secure Socket Layer) nebo TLS (Transport Layer Security).

QUIC Quick UDP Internet Connections je novým transportním protokolem, vyvinutý společností Google, který řeší základní práci s pakety na transportní vrstvě, tak i jejich šifrování a podporu pro protokol HTTP/2.

DoS Denial of Service je útok na odepření internetové služby. Cílem tohoto útoku je znepřístupnit síťové požadavky ostatním klientům.

DDoS Distributed DoS je forma útoku, do kterého je zahrnuto velké množství klientů.

IP IP je označení pro internetový protokol, který slouží pro rozlišení jednotlivých síťových rozhraní pomocí IP adres.

DHCP Dynamic Host Configuration Protocol je protokol aplikační vrstvy, který slouží k automatickému přidělení IP adresy, masky sítě, výchozí brány a IP adresy DNS serveru.

NAT Network Address Translation slouží k překladu IP adres většínou privátní sítě na veřejnou IP adresu.

MPM Mutli-Processing Module - Od verze Apache 2.0, patří multiprocesorový modul k základním funkcím. Tyto moduly jsou zodpovědné za přiřazování síťových portů a příjem požadavků.

HONEYPOT Server, který poskytuje do Internetu běžné webové služby za účelem nalákat útočníky a zastínit tím existenci reálných serverů uvnitř sítě.

Seznam příloh

A Záznamy síťové komunikace jednotlivých typů útoků

A.1 Záznam útoku Slow READ

No.	Time	Source	Destination	Protoco	Length	Info
1	0.00…	10.0.0.3	10.0.0.2	TCP	74	50198 → 80 [SYN] Seq=0 Win=64240 Len=0
2	0.00…	10.0.0.2	10.0.0.3	TCP	74	80 → 50198 [SYN, ACK] Seq=0 Ack=1 Win=6
3	0.00…	10.0.0.3	10.0.0.2	TCP	66	50198 → 80 [ACK] Seq=1 Ack=1 Win=64256
4	0.00…	10.0.0.3	10.0.0.2	HTTP2	105	Magic, SETTINGS[0]
5	0.00…	10.0.0.2	10.0.0.3	TCP	66	80 → 50198 [ACK] Seq=1 Ack=40 Win=65152
6	0.00…	10.0.0.2	10.0.0.3	HTTP2	93	SETTINGS[0]
7	0.00…	10.0.0.2	10.0.0.3	HTTP2	75	SETTINGS[0]
8	0.00…	10.0.0.3	10.0.0.2	TCP	66	50198 → 80 [ACK] Seq=40 Ack=28 Win=6425
9	0.00…	10.0.0.3	10.0.0.2	TCP	66	50198 → 80 [ACK] Seq=40 Ack=37 Win=6425
10	0.00…	10.0.0.3	10.0.0.2	HTTP2	84	HEADERS[1]: GET /
11	0.00…	10.0.0.2	10.0.0.3	TCP	66	80 → 50198 [ACK] Seq=37 Ack=58 Win=6515
12	0.00…	10.0.0.2	10.0.0.3	HTTP2	179	HEADERS[1]: 200 OK
13	0.00…	10.0.0.3	10.0.0.2	TCP	66	50198 → 80 [ACK] Seq=58 Ack=150 Win=642
95	902.…	10.0.0.3	10.0.0.2	TCP	66	50198 → 80 [FIN, ACK] Seq=58 Ack=150 Wi
96	902.…	10.0.0.2	10.0.0.3	TCP	66	80 → 50198 [ACK] Seq=150 Ack=59 Win=651
97	902.…	10.0.0.2	10.0.0.3	TCP	66	80 → 50198 [FIN, ACK] Seq=150 Ack=59 Wi
98	902.…	10.0.0.3	10.0.0.2	TCP	66	50198 → 80 [ACK] Seq=59 Ack=151 Win=642

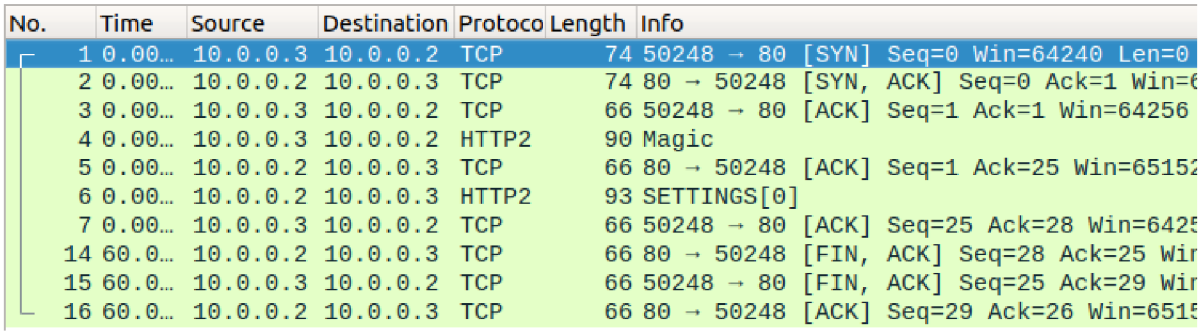
Obr. A.1: Síťová komunikace útoku Slow READ

A.2 Záznam útoku Slow POST

No.	Time	Source	Destination	Protoco	Length	Info
1	0.00…	10.0.0.3	10.0.0.2	TCP	74	50222 → 80 [SYN] Seq=0 Win=64240 Len=0
2	0.00…	10.0.0.2	10.0.0.3	TCP	74	80 → 50222 [SYN, ACK] Seq=0 Ack=1 Win=6
3	0.00…	10.0.0.3	10.0.0.2	TCP	66	50222 → 80 [ACK] Seq=1 Ack=1 Win=64256
4	0.00…	10.0.0.3	10.0.0.2	HTTP2	148	Magic, SETTINGS[0], WINDOW_UPDATE[0]
5	0.00…	10.0.0.2	10.0.0.3	TCP	66	80 → 50222 [ACK] Seq=1 Ack=83 Win=65152
6	0.00…	10.0.0.2	10.0.0.3	HTTP2	93	SETTINGS[0]
7	0.00…	10.0.0.2	10.0.0.3	HTTP2	75	SETTINGS[0]
8	0.00…	10.0.0.3	10.0.0.2	HTTP2	84	HEADERS[1]: POST /
9	0.00…	10.0.0.2	10.0.0.3	TCP	66	80 → 50222 [ACK] Seq=37 Ack=101 Win=651
10	0.00…	10.0.0.3	10.0.0.2	TCP	66	50222 → 80 [ACK] Seq=101 Ack=28 Win=642
11	0.00…	10.0.0.3	10.0.0.2	TCP	66	50222 → 80 [ACK] Seq=101 Ack=37 Win=642
12	0.00…	10.0.0.2	10.0.0.3	HTTP2	236	HEADERS[1]: 200 OK, DATA[1] (text/html)
13	0.00…	10.0.0.3	10.0.0.2	TCP	66	50222 → 80 [ACK] Seq=101 Ack=207 Win=64
296	1381…	10.0.0.3	10.0.0.2	TCP	66	50222 → 80 [FIN, ACK] Seq=101 Ack=207 W
297	1381…	10.0.0.2	10.0.0.3	TCP	66	80 → 50222 [ACK] Seq=207 Ack=102 Win=65

Obr. A.2: Síťová komunikace útoku Slow POST

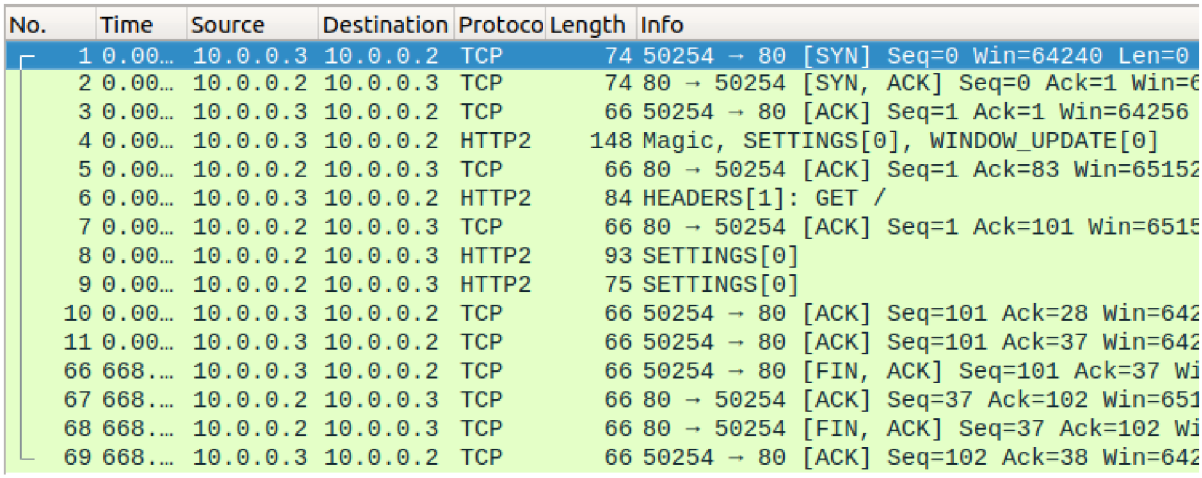
A.3 Záznam útoku Slow PREFACE



No.	Time	Source	Destination	Protoco	Length	Info
1	0.00…	10.0.0.3	10.0.0.2	TCP	74	50248 → 80 [SYN] Seq=0 Win=64240 Len=0
2	0.00…	10.0.0.2	10.0.0.3	TCP	74	80 → 50248 [SYN, ACK] Seq=0 Ack=1 Win=6
3	0.00…	10.0.0.3	10.0.0.2	TCP	66	50248 → 80 [ACK] Seq=1 Ack=1 Win=64256
4	0.00…	10.0.0.3	10.0.0.2	HTTP2	90	Magic
5	0.00…	10.0.0.2	10.0.0.3	TCP	66	80 → 50248 [ACK] Seq=1 Ack=25 Win=65152
6	0.00…	10.0.0.2	10.0.0.3	HTTP2	93	SETTINGS[0]
7	0.00…	10.0.0.3	10.0.0.2	TCP	66	50248 → 80 [ACK] Seq=25 Ack=28 Win=6425
14	60.0…	10.0.0.2	10.0.0.3	TCP	66	80 → 50248 [FIN, ACK] Seq=28 Ack=25 Wir
15	60.0…	10.0.0.3	10.0.0.2	TCP	66	50248 → 80 [FIN, ACK] Seq=25 Ack=29 Wir
16	60.0…	10.0.0.2	10.0.0.3	TCP	66	80 → 50248 [ACK] Seq=29 Ack=26 Win=6515

Obr. A.3: Síťová komunikace útoku Slow PREFACE

A.4 Záznam útoku Slow HEADERS



No.	Time	Source	Destination	Protoco	Length	Info
1	0.00…	10.0.0.3	10.0.0.2	TCP	74	50254 → 80 [SYN] Seq=0 Win=64240 Len=0
2	0.00…	10.0.0.2	10.0.0.3	TCP	74	80 → 50254 [SYN, ACK] Seq=0 Ack=1 Win=6
3	0.00…	10.0.0.3	10.0.0.2	TCP	66	50254 → 80 [ACK] Seq=1 Ack=1 Win=64256
4	0.00…	10.0.0.3	10.0.0.2	HTTP2	148	Magic, SETTINGS[0], WINDOW_UPDATE[0]
5	0.00…	10.0.0.2	10.0.0.3	TCP	66	80 → 50254 [ACK] Seq=1 Ack=83 Win=65152
6	0.00…	10.0.0.3	10.0.0.2	HTTP2	84	HEADERS[1]: GET /
7	0.00…	10.0.0.2	10.0.0.3	TCP	66	80 → 50254 [ACK] Seq=1 Ack=101 Win=6515
8	0.00…	10.0.0.2	10.0.0.3	HTTP2	93	SETTINGS[0]
9	0.00…	10.0.0.2	10.0.0.3	HTTP2	75	SETTINGS[0]
10	0.00…	10.0.0.3	10.0.0.2	TCP	66	50254 → 80 [ACK] Seq=101 Ack=28 Win=642
11	0.00…	10.0.0.3	10.0.0.2	TCP	66	50254 → 80 [ACK] Seq=101 Ack=37 Win=642
66	668.…	10.0.0.3	10.0.0.2	TCP	66	50254 → 80 [FIN, ACK] Seq=101 Ack=37 Wi
67	668.…	10.0.0.2	10.0.0.3	TCP	66	80 → 50254 [ACK] Seq=37 Ack=102 Win=651
68	668.…	10.0.0.2	10.0.0.3	TCP	66	80 → 50254 [FIN, ACK] Seq=37 Ack=102 Wi
69	668.…	10.0.0.3	10.0.0.2	TCP	66	50254 → 80 [ACK] Seq=102 Ack=38 Win=642

Obr. A.4: Síťová komunikace útoku Slow HEADERS

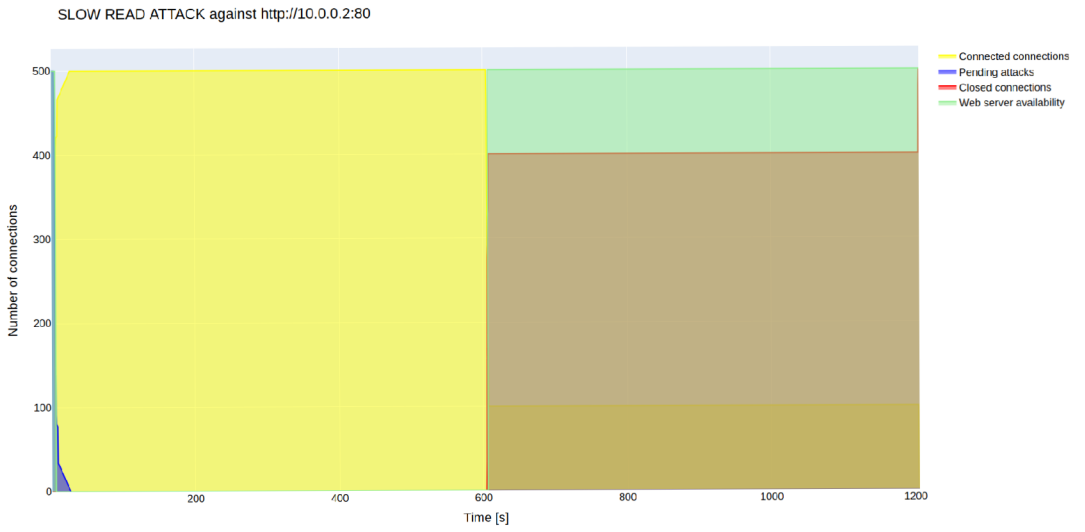
A.5 Záznam útoku Slow SETTINGS

No.	Time	Source	Destination	Protoco	Length	Info
1	0.00…	10.0.0.3	10.0.0.2	TCP	74	50264 → 80 [SYN] Seq=0 Win=64240 Len=0
2	0.00…	10.0.0.2	10.0.0.3	TCP	74	80 → 50264 [SYN, ACK] Seq=0 Ack=1 Win=6
3	0.00…	10.0.0.3	10.0.0.2	TCP	66	50264 → 80 [ACK] Seq=1 Ack=1 Win=64256
4	0.00…	10.0.0.3	10.0.0.2	HTTP2	148	Magic, SETTINGS[0], WINDOW_UPDATE[0]
5	0.00…	10.0.0.2	10.0.0.3	TCP	66	80 → 50264 [ACK] Seq=1 Ack=83 Win=65152
6	0.00…	10.0.0.2	10.0.0.3	HTTP2	93	SETTINGS[0]
7	0.00…	10.0.0.2	10.0.0.3	HTTP2	75	SETTINGS[0]
8	0.00…	10.0.0.3	10.0.0.2	TCP	66	50264 → 80 [ACK] Seq=83 Ack=28 Win=6425
9	0.00…	10.0.0.3	10.0.0.2	TCP	66	50264 → 80 [ACK] Seq=83 Ack=37 Win=6425
10	0.00…	10.0.0.3	10.0.0.2	HTTP2	84	HEADERS[1]: GET /
11	0.00…	10.0.0.2	10.0.0.3	TCP	66	80 → 50264 [ACK] Seq=37 Ack=101 Win=651
12	0.00…	10.0.0.2	10.0.0.3	HTTP2	236	HEADERS[1]: 200 OK, DATA[1] (text/html)
13	0.00…	10.0.0.3	10.0.0.2	TCP	66	50264 → 80 [ACK] Seq=101 Ack=207 Win=64
35	60.0…	10.0.0.2	10.0.0.3	TCP	66	80 → 50264 [FIN, ACK] Seq=207 Ack=101 W
36	60.0…	10.0.0.3	10.0.0.2	TCP	66	50264 → 80 [FIN, ACK] Seq=101 Ack=208 W
37	60.0…	10.0.0.2	10.0.0.3	TCP	66	80 → 50264 [ACK] Seq=208 Ack=102 Win=65

Obr. A.5: Síťová komunikace útoku Slow SETTINGS

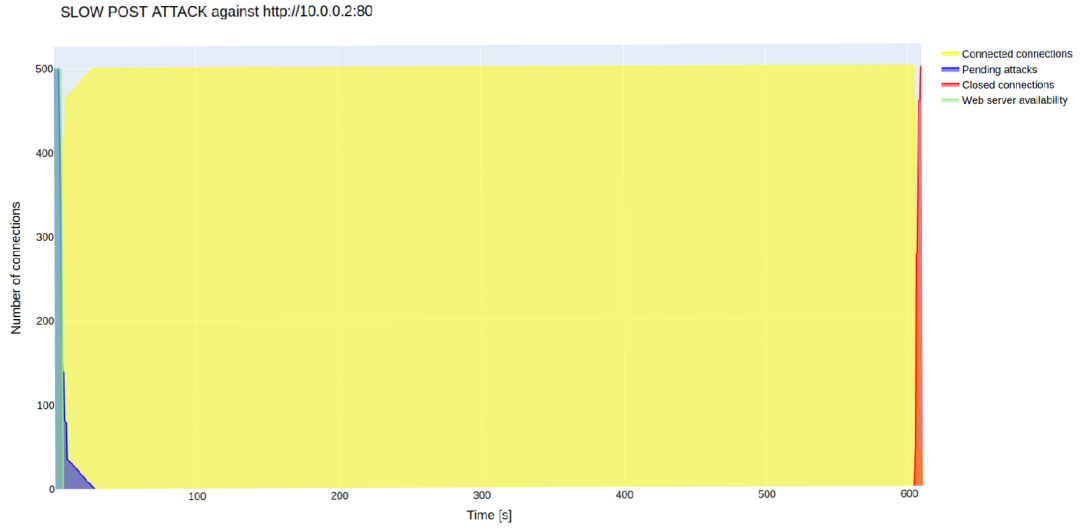
B Generování útoků Slow DoS

B.1 Graf průběhů útoků Slow READ



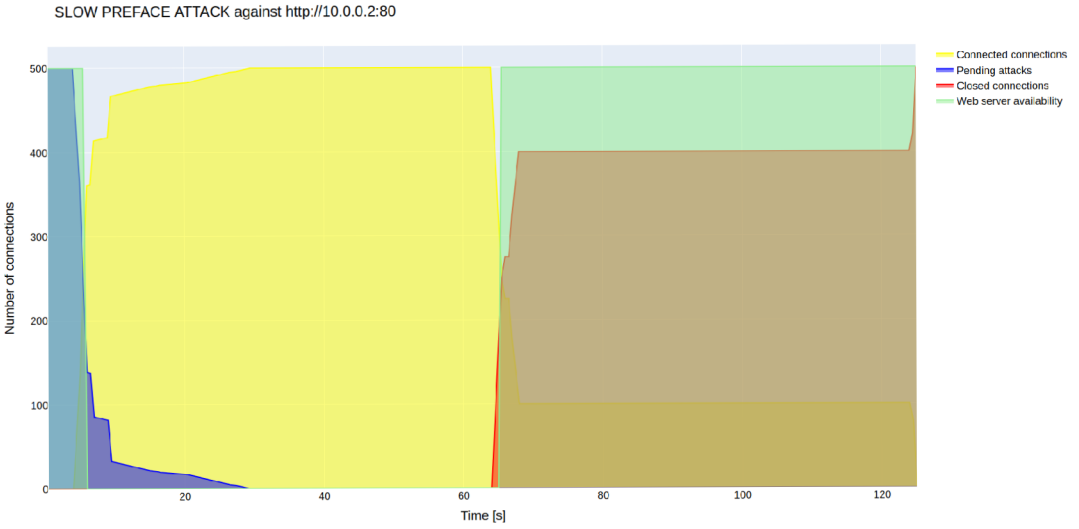
Obr. B.1: Graf vygenerovaných útoků Slow READ generátorem `slowhttp2test.py`

B.2 Graf průběhů útoků Slow POST



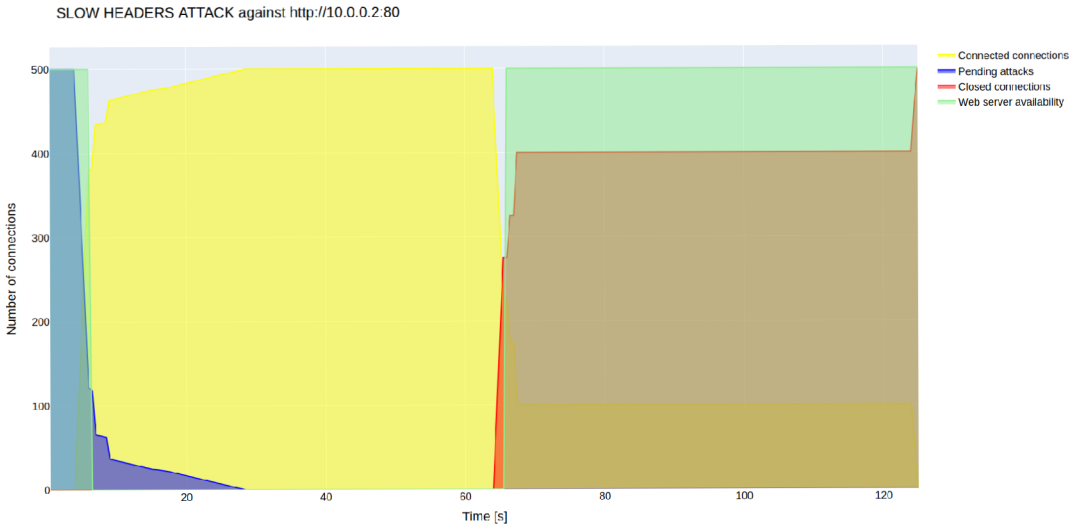
Obr. B.2: Graf vygenerovaných útoků Slow POST generátorem `slowhttp2test.py`

B.3 Graf průběhů útoků Slow PREFACE



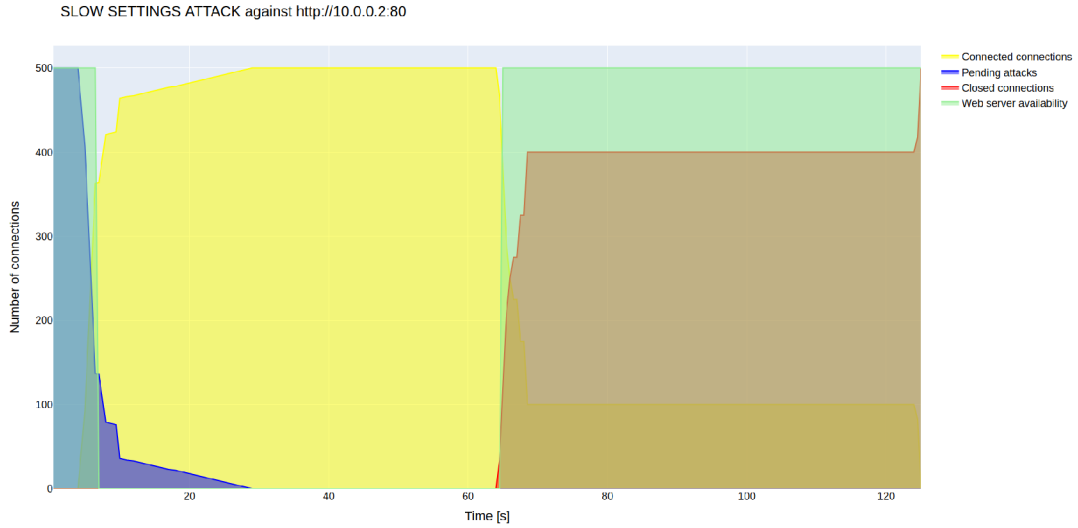
Obr. B.3: Graf vygenerovaných útoků Slow PREFACE generátorem `slowhttp2test.py`

B.4 Graf průběhů útoků Slow HEADERS



Obr. B.4: Graf vygenerovaných útoků Slow HEADERS generátorem `slowhttp2test.py`

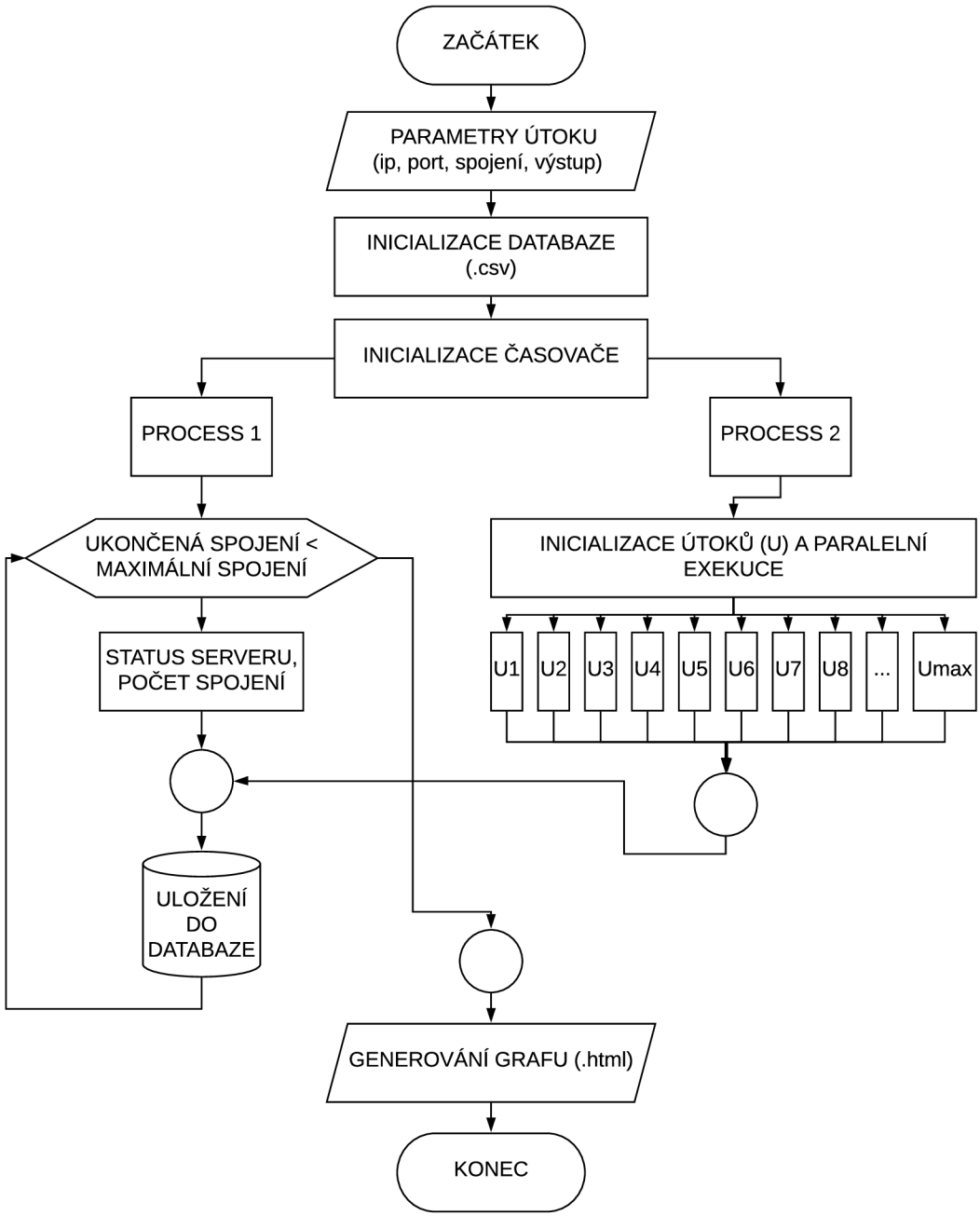
B.5 Graf průběhů útoků Slow SETTINGS



Obr. B.5: Graf vygenerovaných útoků Slow SETTINGS generátorem `slowhttp2test.py`

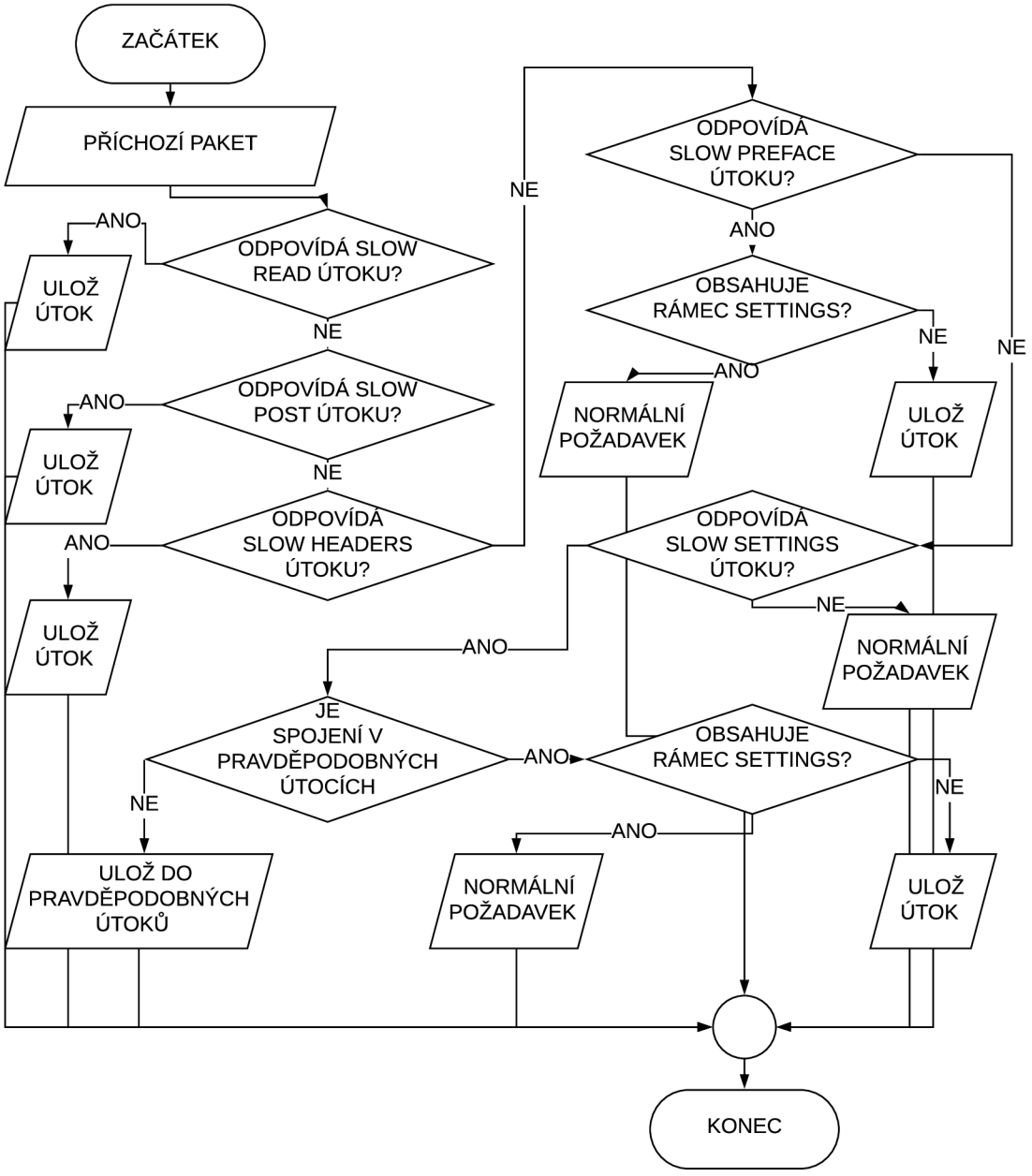
C Vývojové diagramy

C.1 Algoritmus programu `slowhttp2test.py`



Obr. C.1: Vývojový diagram generátoru Slow DoS útoků

C.2 Algoritmus detekce Slow DoS útoků programu `slowhttp2detect.py`



Obr. C.2: Vývojový diagram detektoru Slow DoS útoků

D Obsah přiloženého CD

V příloze této práce se nachází adresář `/slowhttp2test/`, který obsahuje veškeré soubory nutné k úspěšnému provedení Slow DoS útoků. Lze jej taktéž nalézt v online repozitáři `https://github.com/Michael-Jurek/slowhttp2test`.

Dále se v adresáři `/config/` nacházejí konfigurační soubory jednotlivých verzí webových serverů. Obahují sekvenčně uspořádané závislosti programů potřebných pro úspěšné sestavení.

V adresáři `/slowhttp2detect/` se nachází soubory nutné k úspěšné detekci Slow DoS útoků. Tyto soubory se taky vyskytují v online repozitáři `https://github.com/Michael-Jurek/slowhttp2detect`.

- `/` ..... kořenový adresář přiloženého CD
  - `slowhttp2test/` ..... adresář se soubory pro generátor útoků
    - `slowhttp2settings.py` ..... soubor s nástroji umožňujícími provedení útoků
    - `slowhttp2attack.py` ..... soubor s definicemi typů útoků
    - `slowhttp2test.py` ..... hlavní program pro testování útoků
    - `requirements.txt`
    - `README.md`
    - `attack.png`
    - `LICENSE`
  - `config/` ..... adresář se soubory s konfigurací
    - `apache2_4_17.sh`
    - `apache2_4_29.sh`
    - `apache2_4_41.sh`
    - `nginx.sh`
    - `client.sh`
  - `slowhttp2detect/`
    - `slowhttp2detect.py` ..... adresář se soubory pro detekci útoků
    - `requirements.txt`
    - `detect1.png` ..... záznam detekce útoků ze souboru `test_capture1.pcap`
    - `detect2.png` ..... záznam online detekce útoků
    - `test_capture.pcap` .... soubor obsahující zachycenou síťovou komunikaci pro detekci útoků
    - `README.md`
    - `LICENSE`