VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2020 Dominik Richter

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

SLOW RATE DOS ÚTOKY NEZÁVISLÉ NA PROTOKOLU APLIKAČNÍ VRSTVY
SLOW RATE DOS ATTACKS INDEPENDENT OF APPLICATION LAYER PROTOCOL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Dominik Richter**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Marek Sikora**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Dominik Richter ***ID:*** 201350
Ročník: 3 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Slow rate DoS útoky nezávislé na protokolu aplikační vrstvy

POKYNY PRO VYPRACOVÁNÍ:

Slow DoS je specifická a poměrně nová skupina útoků s odepřením služby, způsobující nedostupnost síťových služeb, nejčastěji webových stránek. Slow DoS útoky se vyznačují velmi malým provozem a velkou podobností s legitimním provozem běžných uživatelů, díky čemuž jsou velmi efektivní a obtížně odhalitelné. Obvykle jsou tyto útoky zaměřeny na konkrétní protokoly aplikační vrstvy (např. Slowloris na HTTP protokol), avšak existuje i skupina útoků, jejichž princip lze použít na jakýkoliv protokol aplikační vrstvy. Tyto útoky mají zpravidla mnohem vyšší účinnost a hůře se identifikují. Mezi tyto útoky se řadí např. Slowcomm a SlowNext a SlowReq.
Cílem této práce je analyzovat a popsat minimálně tři výše zmíněné útoky, navrhnout modely těchto útoků a vytvořit testovací prostředí s funkčním generátorem útoků. Dalším cílem práce je vytvořit software, jenž bude schopen tyto útoky v síti detekovat.

DOPORUČENÁ LITERATURA:

[1] CAMBIASO, Enrico, Gianluca PAPALEO, Giovanni CHIOLA a Maurizio AIELLO. Designing and Modeling the Slow Next DoS Attack. International Joint Conference [online]. DOI: 10.1007/978-3-319-19713-5_22.

[2] AIELLO, Maurizio, Gianluca PAPALEO a Enrico CAMBIASO. SlowReq: A Weapon for Cyberwarfare Operations. Characteristics, Limits, Performance, Remediations. International Joint Conference SOCO'13-CISIS'13-ICEUTE'13 [online]. DOI: 10.1007/978-3-319-01854-6_55.

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 je zaměřena na vývoj generátoru útoků typu Slow DoS nezávislých na protokolu aplikační vrstvy a systému schopného tyto útoky detekovat. Tyto útoky se vyznačují využitím velmi malé šířky pásma a podobností s legitimním provozem uživatelů na síti. Díky tomu jsou velmi efektivní a těžce odhalitelné. Navíc se mohou použít na více protokolů aplikační vrstvy modelu ISO/OSI, jako například FTP, SSH nebo HTTP. Konkrétně se práce zabývá útoky Slowcomm, Slow Next a SlowReq. V úvodu práce jsou čtenáři představeny tři protokoly aplikační vrstvy, na kterých se budou implementované útoky prezentovat a testovat. Dále jsou detailněji popsány jednotlivé Slow DoS útoky a postup jejich implementace v testovacím prostředí. Následně byl vytvořen detekční systém IDS, který je schopen detekovat probíhající útok generovaný vytvořeným generátorem. Byla popsána i jeho implementace. Výsledky práce ukazují, že Slow DoS útoky jsou schopny zamezit přístupu k cílové službě rychleji a efektivněji než klasické záplavové útoky. Detekční systém je na druhou stranu schopen je odhalit.

KLÍČOVÁ SLOVA

Pomalé DoS útoky, Slowcomm, Slow Next, SlowReq, Generátor pomalých DoS útoků, IDS, systém detekce průniku

ABSTRACT

This bachelor thesis is focused on the development of a generator of Slow DoS attacks independent of the application layer protocol and a system capable of detecting these attacks. These attacks are characterized by the use of very low bandwidth and similarities to legitimate user traffic on the network. This makes them very effective and difficult to detect. In addition, they can be applied to multiple ISO/OSI application layer protocols, such as FTP, SSH, or HTTP. Specifically, the work deals with Slowcomm, Slow Next and SlowReq attacks. In the introduction, the reader is introduced to three application layer protocols, on which the implemented attacks will be presented and tested. Next, the individual Slow DoS attacks and the procedure of their implementation in the test environment are described in more detail. Subsequently, an IDS detection system was created, which is able to detect the ongoing attack generated by the created generator. Its implementation was also described. The results show that Slow DoS attacks are able to prevent access to the target service faster and more effectively than conventional flood attacks. The detection system, on the other hand, is able to detect them.

KEYWORDS

Slow DoS attacks, Slowcomm, Slow Next, SlowReq, Slow DoS attack generator, IDS, intrusion detection system

RICHTER, Dominik. *Slow rate DoS útoky nezávislé na protokolu aplikační vrstvy*. Brno, 2020, 69 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 „Slow rate DoS útoky nezávislé na protokolu aplikační vrstvy“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Marku Sikorovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Dále bych chtěl poděkovat své přítelkyni a rodině za podporu při studiu a tvorbě práce.

Obsah

Seznam obrázků

Seznam výpisů

Úvod

Bakalářská práce se zabývá poměrně novou skupinou útoků s odepřením služby, které nejčastěji způsobují nedostupnost webových stránek legitimním uživatelům, tzv. pomalými DoS útoky (Slow rate Denial of Services attacks). Vychází z útoků se souhrnným názvem DoS. Tyto útoky, i když jsou známé už několik desetiletí, jsou i nadále velice účinné a poměrně často využívané v jejich sofistikovanějších verzích. Například v roce 2009 byl takovýto útok efektivně použit při prezidentských volbách v Íránu [1]. Je tak nutné stále sledovat jejich vývoj a vytvářet bezpečnější a robustnější systémy pro zajištění kybernetické bezpečnosti.

Společným cílem všech pomalých DoS útoků je zamezení přístupu k určité službě pomocí zasílání relativně malého množství dat v průběhu uměle prodloužené doby přenosu v rámci spojení klient–server. Tímto způsobem může být útok proveden i z pozice méně kvalitně vybaveného útočníka, protože na rozdíl od jiných DoS útoků stačí útok vést i z jednoho počítače s omezeným internetovým připojením a zdroji. Obvykle jsou tyto útoky zaměřeny na konkrétní protokol, např. Slowloris na HTTP (Hypertext Transfer Protocol), avšak existuje i skupina pomalých DoS útoků, které mohou být použity na jakýkoliv protokol aplikační vrstvy. Právě těmi se bude tato bakalářská práce zabývat [2].

Tyto útoky mají zpravidla mnohem vyšší účinnost a hůře se identifikují. Přestože existuje velké množství útoků patřících do této kategorie, stále se tato oblast pokládá za dostatečně neprozkoumanou, zejména kvůli jejímu mládí. Mezi tyto útoky se řadí např. Slowcomm, SlowNext a SlowReq, na které bude tato bakalářská práce zaměřena [3] [4] [5].

Cílem této bakalářské práce je popis všech výše zmíněných pomalých DoS útoků a vytvoření jejich modelů. Následně vytvořit testovací prostředí s funkčním generátorem těchto útoků a otestování jejich účinků na vybraných serverech. Výsledkem je generátor Slow DoS útoků, schopný minimálně zpomalit nebo zamezit přístupu ke službám testovaných serverů. Případně mohou tyto útoky vést k úplnému kolapsu serveru.

Zejména proto, že jsou pomalé DoS útoky tak účinné a méně známé, je třeba věnovat pozornost i obraně proti nim. Druhým cílem této práce je vytvoření systému, schopného tyto útoky v síti detekovat na základě znalostí získaných při jejich zkoumání v první části práce.

První kapitola práce je zaměřena na tři protokoly aplikační vrstvy a jejich popis. Jedná se o protokol HTTP – tento protokol bude využit k demonstraci a testování všech pomalých DoS útoků v této práci, a je proto důležité mu porozumět. Dalšími dvěma protokoly jsou FTP (File Transfer Protocol) a SSH (Secure Shell) ve formě služby OpenSSH. Na nich bude taktéž probíhat testování.

Druhá kapitola teoreticky rozebírá problematiku útoků DoS a popisuje jejich dělení do jednotlivých skupin. Následně je vysvětlena struktura Slow DoS útoků i jejich rozdělení až po útoky nezávislé na aplikační vrstvě, kterými se tato práce zabývá. Hlavní důraz je kladen na zařazení útoků Slow DoS a jejich funkčnost, konkrétně útoků Slowcomm, Slow Next a SlowReq.

Třetí kapitola se zaměřuje na návrh modelů jednotlivých útoků, podle kterých proběhne implementace do komplexního nástroje generátoru pomalých DoS útoků. Zároveň jsou zde představeny bezpečnostní moduly testovaného webového serveru Apache.

Čtvrtá kapitola popisuje implementaci samotného generátoru pomalých DoS útoků a rozdíly při implementaci jednotlivých z nich.

Pátá a šestá kapitola se věnují druhému cíli bakalářské práce – systému schopného zkoumané útoky v síti detekovat. Rozebírá možnosti detekce, které dnešní doba nabízí, dále signatury, podle kterých je detekční systém schopen útoky zaznamenat a zabývá se také popisem implementace a funkčností detekčního systému jako takového.

Sedmá kapitola se zabývá návrhem testovacího prostředí a všech jeho součástí. Popisuje vybrané použité servery, stroj útočníka a oběti, jejich konfigurace a zapojení do interní sítě.

V osmé kapitole práce je popsáno testování generátoru pomalých DoS útoků. Je zde zkoumán vliv útoků na vybrané servery a provedena optimalizace konfigurace útoků vzhledem ke zranitelnostem serverů bez i s použitím bezpečnostních modulů.

V poslední kapitole jsou komentovány výsledky navrženého systému detekce průniku proti generátoru pomalých DoS útoků. Byla zde provedena i simulace jejich distribuované formy.

1 Využívané protokoly aplikační vrstvy

V této kapitole jsou představeny tři protokoly aplikační vrstvy referenčního modelu ISO/OSI, které byly využity při demonstraci a testování pomalých DoS útoků.

1.1 Protokol HTTP

Hypertext Transfer Protocol (dále jen HTTP) je protokol aplikační vrstvy určený pro komunikaci mezi klientem (webovým prohlížečem) s webovým serverem pomocí odesílaných a přijímaných hypertextových dokumentů. Komunikace se serverem probíhá na principu dotaz–odpověď přes TCP (Transmission Control Protocol), který vytvoří spojení, většinou na portu 80. Následně probíhá komunikace pomocí HTTP metod. Protokol HTTP jako takový nezaručuje integritu a šifrování dat. Proto se v dnešní době pro zabezpečení používá šifrované TLS spojení jako nadstavba protokolu TCP. Takto použitá implementace se označuje jako HTTPS [6].

Pro úplný dotaz nebo odpověď jsou nutné určité parametry. Metoda protokolu, URL (Uniform Resource Locator neboli jednotná adresa zdroje) požadovaného objektu, verze protokolu a další nastavení v podobě HTTP hlaviček. Následovat může tělo dotazu. Mezi HTTP metody patří GET, POST, PUT, DELETE, HEAD, OPTIONS a další. Nejvyužívanější jsou první dvě zmíněné [6].

Metoda GET slouží k vyžádání dat (html soubor, obrázek, webové stránky) z webového serveru, nikoli k jejich modifikaci. Je tomu tak z toho důvodu, že požadavky GET se uchovávají v historii prohlížeče nebo vyrovnávací paměti, a kdyby obsahovaly některá senzitivní data, bylo by poměrně jednoduché taková data zneužít. Přesto lze pomocí něj zaslat serveru informace v URL [6].

Pomocí metody POST klient posílá serveru informace od uživatele. Protože tento požadavek není nikde uchováván jako v případě metody GET, mohou se s její pomocí posílat i senzitivní data, která si žádají větší stupeň ochrany. Často se používá pro odesílání dat z webových formulářů, nahrání souborů a podobně. Jeho velikost je limitována pouze v případě posílání dat přes URL [6].

Metody PUT/DELETE na rozdíl od metod GET a PUT vytváří/mažou daný objekt z webového serveru [6].

Metoda HEAD bývá často využívána před načítáním velkého zdroje, zejména pro kontrolu jeho velikosti, dostupnosti nebo modifikací. Je tomu tak z toho důvodu, že tato metoda nezískává od serveru žádná přímá data, pouze informace o nich (tzv. metadata) v podobě hlaviček HTTP odpovědi [6].

Metoda OPTIONS slouží k popisu možností komunikace se serverem. Klient může specifikovat pro zjištění možností, nebo použít hvězdičku (*) pro možnosti

celého serveru. Může tak například zjistit, které metody může zasílat určitému serveru. Velikost těchto požadavků není žádným způsobem limitována až na výjimku, kdy data posíláme v URL [6].

Všechny parametry požadavku (hlavičky) musí být odděleny sekvencí znaků `\r\n`, odřádkováním, navíc na konci všech hlaviček požadavku musí být znaky `\r\n\r\n` (dvojité odřádkování), které reprezentují konec záhlaví požadavku, za nimi například v případě POST požadavku následují samotná data, např. z webového formuláře [6].

Server podporující HTTP verzi 1.0 vrátí odpověď a spojení ihned uzavře [7]. Pokud ale protokol podporuje verzi 1.1, je jako výchozí nastavena jiná komunikační politika - tzv. perzistentní spojení [6].

1.1.1 Perzistentní spojení

Často také nazývané trvalé nebo Keep–Alive spojení souvisí s návrhem komunikace mezi serverem a klientem použitím pouze jednoho TCP spojení pro posílání více požadavků a ušetření tak síťového provozu, na rozdíl od komunikace více spojeními. Jak již bylo řečeno, HTTP verze 1.1 automaticky předpokládá, že klient zamýšlí vytvořit spojení, které nechce hned ukončit a využít ho k zaslání více požadavků. V tomto případě by tedy měl mít požadavek klienta následující HTTP hlavičku, ale v případě HTTP 1.1 ji server vkládá automaticky. Příklad je uveden pouze pro ilustraci:

`Connection: Keep-Alive\r\n`.

Je vhodné dodat, že je povinná hlavička `Host`, která obsahuje IP adresu serveru a volitelně i port:

`Host: 10.10.0.2\r\n`.

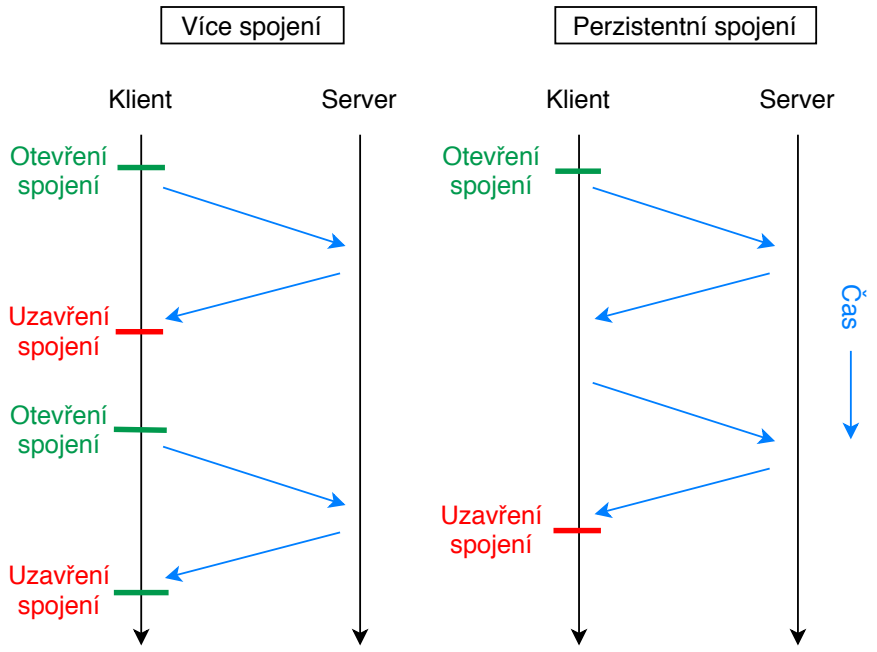
Pokud klient nebo i server chtějí spojení ihned po odeslání požadavku a odpovědi uzavřít, musí naopak použít tuto HTTP hlavičku:

`Connection: Close\r\n`.

Pokud je požadavek rozdělen do více paketů, je také nutné, aby požadavek obsahoval hlavičku `Content-Lenght`. Tato hlavička udává serveru hodnotu velikosti celého požadavku. Server potom ví, kdy mu přišel kompletní požadavek a může s ním tak dále pracovat.

`Content-Lenght: 129\r\n` [6].

Perzistentní spojení se využívá velmi často. Například pokud chce klient načíst webovou stránku s textem a obrázky. V případech, kdy by perzistentní spojení nebylo použito, byl by každý element stránky poslán zvlášť v separátním spojení. Každé takové spojení ovšem vyžaduje nové navázání TCP spojení, protože se po dokončení požadavku uzavře. Využití trvalého spojení tak šetří síťový provoz, snižuje latenci jednotlivých požadavků a redukuje využití procesoru (obr. 1.1). Proto je od HTTP verze 1.1 tato politika nastavena jako výchozí. V podstatě jediná nevýhoda této metody je trvalé obsazení spojení serveru, které blokuje připojení dalších klientů. Tohoto faktu hojně využívají například pomalé DoS útoky [8].



Obr. 1.1: Rozdíl mezi perzistentním a více spojeními

Příklad požadavku klienta a odpověď serveru pomocí protokolu HTTP verze 1.1 s využitím metody GET může vypadat například takto. Za hlavičkami HTTP odpovědi server posílá i tělo odpovědi (např. HTML kód webové stránky).

Požadavek klienta:

```
GET / HTTP/1.1\r\n
Host: [...]\r\n
User-Agent: Mozilla/5.0 [...]\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-US,en;q=0.5\r\n
Accept-Encoding: gzip, deflate\r\n
Connection: keep-alive\r\n\r\n
```

Odpověď serveru:

```
HTTP/1.1 200 OK
Date: Mon, 25 Nov 2019 19:31:28 GMT\r\n
Server: Apache/2.4.41 (Debian)\r\n
Last-Modified: Tue, 27 Aug 2019 10:39:13 GMT\r\n
Accept-Ranges: bytes\r\n
Vary: Accept-Encoding\r\n
Content-Length: 10701\r\n
Keep-Alive: timeout=5, max=100\r\n
Content-Type: text/html\r\n\r\n
```

1.2 Protokol FTP

Protokol File Transfer Protocol (dále jen FTP) je síťový protokol aplikační vrstvy referenčního modelu ISO/OSI, který zprostředkovává přenos souborů mezi klientem a serverem. Uživatelé (klienti) se mohou připojovat pomocí implementovaného přihlašovacího protokolu, ale mohou se přihlásit i jako anonymní uživatelé, pokud je k tomu server využívající tento protokol nakonfigurován [9].

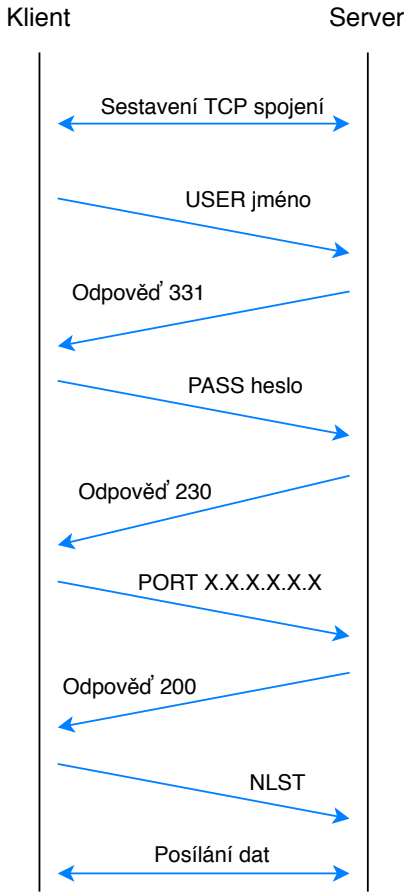
Využívá porty 21 a 20. Port 21 pro řízení spojení pomocí FTP příkazů, jako například USER, PASS, PORT, NLST, QUIT a další, přičemž kompletní příkaz je vždy oddělen znaky `\r\n`. Port 20 slouží k vlastnímu přenosu dat. V této bakalářské práci byl využit k testování nezávislosti útoků na protokolech aplikační vrstvy [9].

Příkazy USER a PASS se používají při autentizaci klienta pro přenos uživatelského jména a hesla. Příkaz PORT specifikuje port, na kterém klient naslouchá a příkaz NLST potom vrací jmenný seznam adresáře, případně souboru. Jde o podobný příkaz k příkazu LIST, který kromě jmen vrací i další informace ohledně souborů a adresářů. QUIT slouží k ukončení spojení [9].

FTP jako takový funguje ve výchozím nastavení nezabezpečeně. Přihlašovací údaje (jméno a heslo) jsou po síti přenášeny v nijak nešifrované textové podobě. To může vést ke snadnému odchycení komunikace útočníkem, odhalení uživatelského jména a hesla a například úniku důležitých dat z FTP serveru. Pro zabezpečené spojení, které zajistí důvěrnost přihlašovacích údajů, je FTP často kombinováno s protokolem SSL (Secure Sockets Layer)/TLS (Transport Layer Security). Tyto kryptografické protokoly poskytují zabezpečené spojení v síti Internet. Tato kombinace se označuje jako FTPS [9].

Jak je vidět na obrázku 1.2, v rámci protokolu FTP je nejdříve sestaveno TCP spojení jako v případě protokolu HTTP, potom probíhá autentizace uživatele pomocí již zmíněných metod USER a PASS. Server příslušnou odpovědí potvrdí příjem

a uživatele přihlásí. Příkaz PORT je poslán klientem k zahájení datového spojení vyžadovaného pro přenos dat mezi klientem a serverem. Poté už je proveden pouze příkaz NLST a spojení je navázáno. Následuje samotný přenos dat mezi serverem a klientem.



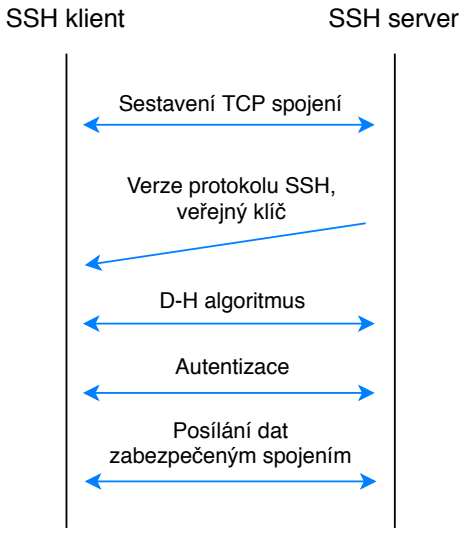
Obr. 1.2: Komunikace v rámci protokolu FTP

1.3 Protokol SSH

Secure Shell (dále jen SSH) je kryptografický síťový protokol aplikační vrstvy referenčního modelu ISO/OSI, který vytváří bezpečný kanál přes nezabezpečenou síť v architektuře klient-server, spojující klientskou aplikaci SSH se serverem SSH. Typickými příklady použití jsou vzdálené přihlášení a provádění příkazů. Využívá port 22 [10].

SSH byl navržen Tatu Ylonenenm jako náhrada za protokol Telnet a za další nezabezpečené protokoly vzdáleného připojení. Telnet je síťový protokol aplikační

vrstvy referenčního modelu ISO/OSI využívající port 23, zajišťující připojení ke vzdálenému počítači pomocí textového uživatelského rozhraní. Oproti protokolu SSH ovšem přenášená data nejsou šifrována a pro útočníka je tak velice jednoduché zachytit probíhající výměnu dat. Kvůli této skutečnosti se dnes takřka nepoužívá. Účelem šifrování používaného SSH je tak zajistit důvěrnost a integritu dat v nezabezpečené síti, jako je Internet [10].



Obr. 1.3: Sestavení zabezpečeného spojení pomocí protokolu SSH

Protokol SSH používá šifrování k zabezpečení spojení mezi klientem a serverem, přičemž spojení iniciuje a řídí klient. Server pouze potvrzuje a ověřuje parametry zasílané klientem. Autentizace uživatelů, zadané příkazy a přenosy souborů jsou šifrovány, aby byly chráněny před útoky v síti. Využívá se zde infrastruktury správy a distribuce veřejných klíčů z asymetrické kryptografie (PKI - Public Key Infrastructure). Jak je vidět na obrázku 1.3, průběh protokolu je následující. SSH klient naváže TCP spojení se serverem, který odpoví zprávou s verzí protokolu, kterou podporuje. Pokud klient jednu z poskytnutých verzí rovněž podporuje, spojení pokračuje. Server v odpovědi posílá i svůj veřejný klíč, pomocí něhož si klient ověří jeho identitu svým privátním klíčem. Následně se obě strany dohodnou na klíči pro probíhající spojení pomocí Diffie–Hellmanova algoritmu, který zajistí šifrování celé relace [10].

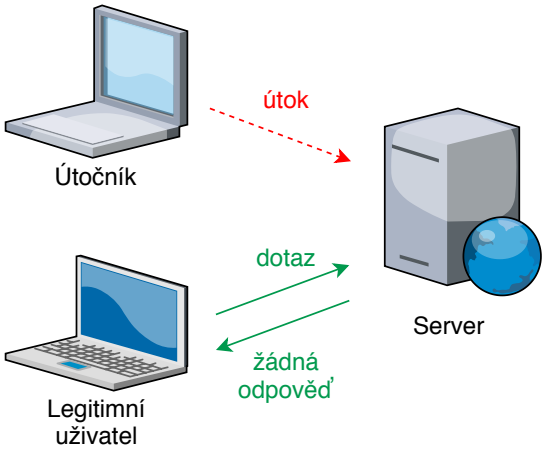
Po této fázi následuje autentizace uživatele. Nejčastěji se využívá autentizace pomocí symetrického předsdíleného klíče neboli hesla zadávaného klientem. Druhou doporučovanou možností je použití asymetrických SSH klíčů – veřejného pro šifrování dat a soukromého pro jejich dešifrování. V tomto případě klient posílá ID dvojice

klíčů, které chce při autentizaci použít. Server si následně ověří, že ve svém seznamu autorizovaných klíčů má veřejný klíč s daným ID. Pokud ano, vygeneruje náhodné číslo a použije tento veřejný klíč pro jeho zašifrování. Zašifrovanou zprávu pošle klientovi, který, pokud vlastní odpovídající soukromý klíč, je schopen zprávu dešifrovat. Klient poté vytvoří ze získané zprávy spolu s dříve vygenerovaným klíčem relace hash pomocí funkce SHA-1 a výsledek zašle serveru, který provede stejnou operaci a porovná výsledky své a přijaté hašovací funkce. Pokud jsou výsledky totožné, tak je prokázáno, že klient vlastní unikátní soukromý klíč jako dvojici k veřejnému klíči a je autentizován [10].

2 DoS útoky

V dnešní době se síť Internet považuje za takřka nenahraditelný komunikační prostředek propojující několik milionů počítačů a jiných zařízení na světě a poskytuje jim požadované služby a informace. Protože ovlivňuje každodenní život člověka, servery provozující tyto služby a obsahující tyto informace musejí být chráněny odpovídajícími prostředky. Nevýhoda je v tom, že síť Internet propojuje jak legitimní uživatele, tak i ty nebezpečné, kteří se tak mohou snáze dostat k informacím uloženým tam, kde by správně neměli mít žádný přístup.

Útoky typu DoS se nejčastěji vedou proti webovým serverům a kladou si za cíl znepřístupnit, nebo alespoň zpomalit jejich služby a zatížit servery tak, že už nejsou schopny navazovat spojení s legitimními uživateli (viz obr. 2.1). Kompletní zahlcení zdrojů serveru může vést až k úplnému pádu systému [8].



Obr. 2.1: Princip útoků DoS

Z DoS útoků se později vyvinuly Distribuované Denial of Services útoky (dále jen DDoS), které fungují na stejném principu jako útoky DoS, ovšem k provedení útoku využívají více synchronizovaných počítačů, a jsou tak schopny docílit požadovaného účinku rychleji. Je také mnohem obtížnější je detekovat. Pro útočníka je ale jednodušší infikovat stroje po celém světě, než vlastnit několik počítačů, které by dohromady mohly generovat útoky DDoS. Tyto infikované stroje se nazývají boti nebo zombie a dohromady vytváří tzv. botnet, tedy síť, ze které v závislosti na útočníkovi boti synchronně spouští útok na společný cíl [8].

Zpravidla se DoS útoky dělí na útoky využívající zranitelnost systému a záplavové (flood) útoky. Kromě jiného mohou být dále rozděleny podle síťových vrstev referenčního modelu ISO/OSI a protokolů, kterých se snaží zneužít [8].

Záplavové útoky

Záplavové útoky jsou založeny na generování co největšího počtu požadavků velkou přenosovou rychlostí. Chtějí tedy dosáhnout co největšího toku sítě, aby zahltily výpočetní kapacity cíleného serveru. Požadavky od dalších legitimních uživatelů se tak začnou zahazovat, protože server již nedokáže vytvářet nová spojení a nestihne vygenerovat odpověď v požadovaném čase. Tím útočník zamezí připojení ostatních legitimních uživatelů ke službám serveru. Takovéto útoky jsou jednoduché na implementaci, a proto jsou často používané. V dnešní době, kdy jsou servery připojeny do sítě linkami s gigabitovými rychlostmi, je ovšem úspěšnost takového útoku založena na rychlosti připojení útočníka a spoléhá se i na to, že bude využita varianta DDoS. Jeden počítač by totiž nebyl schopen vygenerovat takové množství provozu. Mezi záplavové útoky se řadí například TCP Flood, UDP Flood nebo Smurf [8].

Útoky využívající zranitelnost systému

Útoky využívající zranitelnost systému (logické útoky) jsou oproti záplavovým útokům účinnější a snáze proveditelné z jednoho počítače. Využívají zranitelností protokolů nebo systémů, na které útočí. Jako příklad může být uveden protokol TCP transportní vrstvy. Při navazování spojení přes tento protokol posílá server po výzvě klienta zprávu s příznakem SYN a ACK, potom čeká na odpověď klienta o potvrzení s příznakem ACK. Pokud ovšem tato odpověď nepřijde, tak server čeká a po určitém čase odešle zprávu znovu. Až po druhém pokusu o navázání spojení server spojení ukončí. Pokud se útočníkovi podaří navázat více spojení současně (využije této zranitelnosti TCP protokolu), dojde k vyčerpání zdrojů serveru a k realizaci útoku DoS, protože server nezvládne odpovídat na další požadavky legitimních uživatelů, které budou zahazovány. Mezi logické útoky patří například výše diskutovaný TCP SYN attack, Ping of Death nebo Land attack [8].

2.1 Pomalé DoS útoky

Postupně se ale vyvinula další kategorie útoků DoS – pomalé DoS útoky, které se používají pouze na protokoly aplikační vrstvy referenčního modelu ISO/OSI. Výhoda spočívá v tom, že zatímco většina DoS útoků pracujících na transportní vrstvě zahrnuje odesílání velkého množství dat, aby zahltily šířku pásma oběti, v případě útoků na protokoly aplikační vrstvy není tak velká výměna dat potřeba, protože počet spojení, který je aplikační vrstva schopna poskytnout, je výrazně nižší než na vrstvě transportní. Proto mohou být provedeny pouze z jednoho počítače, případně v dnešní době z mobilního telefonu nebo smart či IoT zařízení. Tato kategorie může být zařazena mezi záplavové útoky a útoky využívající zranitelnost systému současně. Vyznačují se velmi malým provozem a velkou podobností s legitimním

provozem běžných uživatelů, díky čemuž jsou velmi efektivní a obtížně odhalitelné klasickými detekčními systémy [2].

Strategie pomalých DoS útoků se skládá z různých kroků, vedoucích k vyčerpání zdrojů serveru s využitím malé šířky pásma Nejdříve se vytvoří maximální počet spojení se serverem. Spojení jsou udržována obsazená, ale nevyužitá, pomocí perzistentních spojení, aby byl přenos co nejdéle zdržen a server neměl kapacitu pro vytvoření dalších spojení. Perzistentní spojení se ustanoví pomocí časového limitu, tzv. timeoutu, který má každý server definován v konfiguračních souborech. Jde o časový úsek, ve kterém server čeká na odpověď klienta. Pokud klient neodpoví, po uplynutí této doby server spojení uzavře. Proto je nutné posílat před vypršením časového úseku tzv. udržovací požadavky, které zajistí, že server spojení neuzavře [2].

Při výpočtu tohoto časového úseku útočníkem je třeba snížit serverem udávanou hodnotu o časové zpoždění při odeslání paketu mezi strojem útočníka a serverem a o dobu nutnou pro zpracování požadavku. Pokud je udržovací požadavek zaslán včas, dojde k resetování timeoutu a spojení zůstane otevřené. Proces se dále opakuje v cyklu. Po úspěšném navázaní a udržení co největšího počtu spojení ale některá mohou zůstat nevyužita, nebo je server z nějakého důvodu uzavře. Algoritmus útoku se proto snaží navázat spojení znovu. Proto legitimní uživatel nemá moc šancí navázat vlastní spojení.

Tímto způsobem dojde postupně k obsazení všech možných spojení (zaplnění front serveru), což vede k tomu, že požadavky legitimních uživatelů jsou zahazovány, nebo je server nedokáže obsloužit v požadovaném čase. Tím dojde k zablokování či minimálně ke zpomalení dotazované služby. Na rozdíl od záplavových DoS útoků, kde je velké množství odeslaných požadavků nepotřebných, u pomalých DoS útoků v podstatě každý paket hraje svou roli. Díky tomu, že se posílá malé množství dat pomalou rychlostí (pod 1 KB/s), jsou tyto škodlivé pakety velmi jednoduše zaměňovány s normálním síťovým provozem uživatelů s pomalým připojením k síti.

Pomalé DoS útoky se dále dělí na útoky s čekajícími požadavky, útoky s dlouho trvající odpovědí a útoky vícevrstvé [2].

Útoky s čekajícími požadavky

Útoky s čekajícími požadavky jsou charakteristické posíláním nekompletních požadavků na server, za účelem obsazení všech jeho zdrojů, a dosáhnout tak zamezení služby. Například v případě protokolu HTTP využívají jedné jeho vlastnosti, která umí posílaná data rozdělit do několika TCP segmentů. Server je tím donucen čekat na dokončení přijímaného požadavku, ke kterému ale většinou vůbec nedojde. Útočník takto obsadí maximální počet spojení poskytovaných serverem, které stále udržuje. Server tak nemá prostředky k obsloužení legitimních uživatelů. Příkladem mohou být útoky Slowloris [2].

Útoky s dlouho trvající odpovědí

Útoky s dlouho trvající odpovědí jsou na druhou stranu charakteristické posíláním kompletních validních požadavků serveru pomocí zmíněných perzistentních spojení, často na větší soubory (obrázky), které nutí server k náročnějším výpočtům a sama odpověď potom trvá déle. Server na tento požadavek legitimně odpoví, ovšem útočník čte data po velmi malých částech. Tím útočník dlouhodobě blokuje dostupná TCP spojení. Tuto techniku využívá například útok Slow Read [2].

Vícevrstvé útoky

Vícevrstvé DoS útoky sdílejí typickou nízkou přenosovou rychlost útoků aplikační vrstvy. Navíc využívají transportní vrstvu k dosažení pomalé odpovědi serveru [2].

V kategorii pomalých DoS útoků se ještě více vyčleňují útoky nezávislé na jednotlivých protokolech aplikační vrstvy, kterými se tato práce bude zabývat. Lze je použít například na protokoly HTTP, SMTP nebo FTP zároveň bez výraznějších změn v implementaci [2]. Patří mezi ně již zmíněné útoky Slowcomm, Slow Next a SlowReq [3] [4] [5].

2.1.1 Slowcomm

Útok Slowcomm posílá velké množství pomalých nevalidních požadavků (tzn. rychlostí menší než 1 KB/s) na server, nutící ho k vyčerpání volných spojení na aplikační vrstvě při čekání na ukončení požadavku, které ale nikdy neproběhne [3]. Např. protokol HTTP má za záhlavím uzavírací sekvenci znaků `\r\n\r\n`, tedy dvojité odřádkování. Útočník tyto znaky nikdy nepošle, čímž nutí server nekonečně čekat na dokončení požadavku klienta (útočníka) [6].

Slowcomm je strukturován do tří částí. Nejdříve se ustanoví maximální počet perzistentních spojení s cíleným serverem. Spojení potom zůstávají obsazená, ovšem bez datového provozu, tzn. nevyužitá. Až v bodě, kdy dochází k dovršení timeoutu, jsou v jednotlivých spojeních odesílány již zmíněné udržovací požadavky, které timeout resetují. Tyto požadavky jsou ve skutečnosti jen další nekompletní požadavky. Z podstaty věci by tak v rámci tohoto útoku měl být payload (tzn. obsah požadavku útočníka) univerzální při použití na jakýkoliv protokol aplikační vrstvy, protože server požadavek kontroluje a vykonává až po obdržení kompletního požadavku. Toto tvrzení se následně potvrdilo při testování v kapitole 8.3. Server je tedy opět donucen čekat na doručení kompletního požadavku. Následně je před další expirací timeoutu poslán nový udržovací požadavek, který je znovu nedokončený. Toto se samozřejmě i nadále opakuje až do dosažení efektu útoku DoS. V dalším kroku se snaží uzavřená spojení ihned obnovovat. Obsahuje softwarovou komponentu, která

kontroluje, zda jsou všechna otevřená spojení stále udržována. Pokud ne, ihned se je snaží obnovit. Legitimní uživatel má tak znovu mnohem menší šanci na navázání spojení [3].

Slowcomm se ukazuje jako sofistikovanější v porovnání s útokem Slowloris, neboť potřebuje menší šířku pásma k dosažení DoS útoku a je schopen zaútočit na více protokolů aplikační vrstvy pomocí stejného payloadu (v tomto případě jednoho znaku) v udržovacím požadavku [3].

Útočník tedy tímto způsobem může dosáhnout útoku DoS s průměrnými technickými zdroji, protože využívá malou šířku pásma. Jako ostatně všechny pomalé DoS útoky, nemusí být použit pouze na protokol HTTP. Navíc je poměrně těžké takovýto útok zachytit, protože logy (soubory pro záznam určitých činností) jsou nejčastěji aktualizovány pouze pokud je přijat kompletní požadavek. Odhalit tento útok je tak v podstatě možné až po jeho ukončení, kdy například podle IP adresy klienta (útočníka) může být nadměrná aktivita jednoho uživatele vyhodnocena jako útok DoS [3].

2.1.2 Slow Next

Tento pomalý DoS útok je založen na posílání validních požadavků serveru pomocí perzistentního spojení na aplikační vrstvě. Za účelem dosažení DoS útoku se snaží obsadit maximální počet spojení současně [4].

Útok zneužívá tzv. parametr next [4], tj. čas mezi koncem odpovědi serveru a začátkem dalšího požadavku klienta v rámci jednoho perzistentního spojení. Každé spojení posílá validní požadavky na server. Server následně legitimně odpoví. Po obdržení celé odpovědi využije útočník timeout pro zpoždění odeslání dalšího validního požadavku již založeným spojením. Tímto způsobem donutí server čekat na další požadavek a neuvolňovat spojení pro legitimní uživatele, dokud timeout nevyprší. Těsně před vypršením timeoutu je ale poslán udržovací požadavek, v tomto případě opět legitimní, následkem čehož dojde k resetování timeoutu. Tento postup se opakuje – útočník znovu čeká na vhodnou dobu pro odeslání udržovacího požadavku s ohledem na timeout [4].

Tímto způsobem útočník inicializuje maximální počet spojení, které je server schopen poskytnout. Zamezí tak ostatním legitimním uživatelům ustanovit další spojení. Jejich požadavky budou proto z důvodů vyčerpané kapacity serveru zahazovány. Dojde tak k útoku DoS a znepřístupnění či minimálně výraznému zpomalení služby legitimním uživatelům, popřípadě může dojít i k totálnímu kolapsu serveru.

Takto definovaný útok se ovšem nemusí použít pouze na protokol HTTP, ale může se použít v podstatě na jakýkoliv protokol aplikační vrstvy. Pouze se bude opakovat princip prodlužování spojení. Je velice efektivní, protože chování takto

navrženého útoku lze jen obtížně zachytit detekčními systémy či firewally, protože odesílání a přijímání požadavků bude považováno za legitimní. Mohlo by se pouze předpokládat, že klient má pomalé připojení k internetu, a proto mu zaslání odpovědi trvá déle [4].

2.1.3 SlowReq

Při analýze tohoto útoku bylo zjištěno, že je v podstatě totožný s útokem Slowcomm. Ze získaných odborných materiálů [5] vyplývá, že při útoku SlowReq znovu dochází k navázání maximálního počtu perzistentních spojení, jejich následné udržení a posílání nevalidních paketů s co možná největším časovým zpožděním (timeoutem) za účelem vyčerpání zdrojů serveru a dosažení jeho nedostupnosti pro legitimní uživatele. Tento princip je naprosto shodný s útokem Slowcomm, proto bude v dalších částech práce tento útok zařazen pod útok Slowcomm a tyto dva útoky považovány za jeden. V rámci implementace budou tedy realizovány pouze dva útoky – Slow Next a Slowcomm.

3 Modely pomalých DoS útoků

Tato kapitola se zaměřuje na popis modelů jednotlivých pomalých DoS útoků. Je zde definována jejich struktura, podle které byly implementovány. Jelikož nebyl nalezen žádný volně dostupný generátor těchto útoků, je pro účel práce nutné útoky implementovat do podoby vlastního generátoru pomalých DoS útoků. Ve druhé části je zdokumentován postup při implementaci generátoru, popsány využité nástroje a vysvětleno použití vytvořeného generátoru.

3.1 Návrh modelů útoků

V této části budou navrženy modely útoků Slow Next a Slowcomm. Pomalý DoS útok SlowReq byl z implementace vyřazen, protože se jeho definice shodovala s definicí útoku Slowcomm (viz kapitola 2.1.3). Pro srozumitelnost jsou modely prezentovány na webovém serveru.

Pro správné fungování všech pomalých DoS útoků je nutné vědět, jaké má webový server nastaveny tzv. timeouty. Těchto timeoutů má server nakonfigurovaných hned několik. Nejdůležitějšími z nich jsou tzv. Timeout a KeepAliveTimeout. Timeout určuje maximální dobu, po kterou webový server udrží otevřené jedno perzistetntní spojení při čekání na doručení zbytku nekompletního požadavku, potom server spojení ukončí. KeepAliveTimeout definuje, jak dlouho bude webový server čekat na další požadavek v rámci jednoho perzistentního spojení. Pomocí něj se určuje doba, po které má útočník poslat tzv. udržovací požadavky, aby server neuzavřel spojení. V případě webového serveru Apache je možné tuto informaci zjistit v konfiguračním souboru

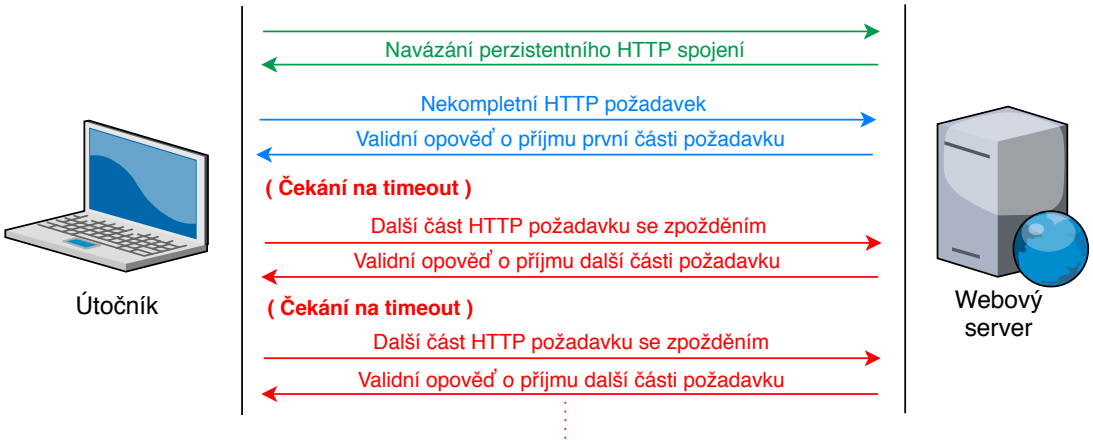
`/etc/apache2/apache2.conf`.

Z tohoto souboru bylo zjištěno, že výchozí hodnota KeepAliveTimeoutu je 5 sekund a hodnota Timeoutu je 300 sekund. Ale i další informace mohou být užitečné. Ze souboru lze zjistit i maximální počet požadavků během jednoho perzistentního spojení, který je zde 100, ale i maximální počet spojení, které server obslouží – 150. Toto omezení naznačuje, kolik spojení musí útočník minimálně navázat, aby bylo možné provést pomalý DoS útok. Zároveň zde je i možnost perzistentní spojení vypnout. Informace o Timeoutu a maximálním počtu požadavků za spojení může být zjištěna i z odpovědi serveru na GET požadavek v hlavičce Keep–Alive.

Další zkoumané protokoly FTP a SSH mají obdobné parametry rovněž definovány ve svých konfiguračních souborech. Více viz kapitoly 8.3 a 8.4.

3.1.1 Slowcomm model

Tento útok posílá webovému serveru požadavky sice pomalou rychlostí, ale nevalidní (nekompletní), opět pomocí perzistentního HTTP spojení, jak bylo popsáno v kapitole 2.1.1. Webový server je proto nucen čekat na dokončení požadavku klienta (útočníka), ke kterému ale nikdy nedojde. Díky udržovacím požadavkům je pak schopen dosáhnout útoku DoS.



Obr. 3.1: Model útoku Slowcomm

Jak je naznačeno na obr. 3.1, nejprve je navázáno TCP spojení mezi útočníkem a webovým serverem. Poté se vytváří maximální počet perzistentních spojení, ve kterých jsou posílány úvodní nekompletní HTTP požadavky HEAD – nejsou zakončeny dvojitým odřádkováním (viz kapitola 1.1). Níže je zobrazen příklad požadavku útočníka.

Požadavek útočníka:

```
HEAD / HTTP/1.1\r\n
```

Aby bylo možné navázat tolik perzistentních spojení z jednoho počítače, musí každé spojení vycházet z jiného portu. Když požadavky vychází pouze z jednoho portu, jsou z logiky věci přidávány do pouze jednoho perzistentního spojení, což splňuje jeho samotný smysl. Proto každé spojení musí být svým způsobem unikátní. To zaručí různá čísla portů náhodně generovaná pro každé perzistentní spojení.

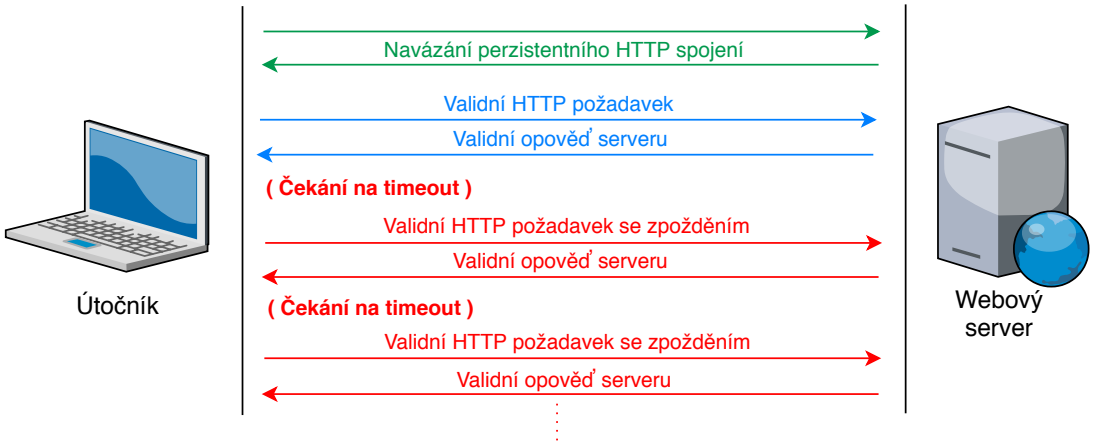
Tímto útočník donutí webový server čekat na zbytek požadavku. Protože jsou odesílány nekompletní požadavky, postačí v záhlaví odeslat pouze metodu a verzi protokolu, není nutná ani hlavička `Host`. Během toho server nijak nedopovídá, pouze posílá zprávu s příznakem ACK o přijetí části požadavku a udržuje spojení. Další části požadavku, které útočník posílá, v tomto případě doslova jedno písmeno „i“ bez

jakýchkoliv HTTP hlaviček, plní roli udržovacích požadavků. Webový server si myslí, že je stále posílán původní požadavek, konkrétně další části záhlaví požadavku.

Webový server Apache ovšem přišel s jednorázovým řešením, které se snaží podobným typům útoků předcházet. Je jím bezpečnostní modul `mod_reqtimeout` [11]. Naštěstí pro útočníka se ale i toto opatření může lehce obejít pomocí obnovy spojení díky udržovacím požadavků, jak bylo popsáno výše, v intervalech odesílání udržovacích požadavků do 10 sekund. Při implementaci byl tento interval (timeout) nastaven na 7 sekund. Ochranné moduly webového serveru Apache budou rozebrány v kapitole 3.2.

3.1.2 Slow Next model

Jak již bylo uvedeno v kapitole 2.1.2, základní princip toho pomalého DoS útoku je posílání validních HTTP požadavků na webový server několika navázanými spojeními zároveň. Jakmile útočník obdrží odpověď webového serveru, využije tzv. timeoutu (viz kapitola 3.1) a odpovídá zpět co nejpozději tak, aby spojení nebylo přerušeno. Pokud se útočníkovi podaří obsadit všechna spojení serveru a udržet je otevřená, způsobí pomalý DoS útok a znepřístupní tak službu (webovou stránku) legitimnímu uživateli.



Obr. 3.2: Model útoku Slow Next

Jak je naznačeno na obr. 3.2, nejprve je nutné sestavit perzistentní spojení mezi útočníkem a webovým serverem, jako v případě útoku Slowcomm. Útočník tedy naváže TCP spojení a odešle úvodní HTTP požadavek HEAD se všemi náležitostmi (minimálními) na IP adresu webového serveru. Jak již bylo zmíněno v kapitole 3.1, perzistentních spojení musí útočník navázat minimálně 150, aby došlo k vyčerpání

zdrojů serveru a útok byl úspěšný. Požadavek HEAD je vhodný kvůli jeho malé spotřebě šířky pásma, protože s odpovědí serveru vrací pouze HTTP hlavičky týkající se dotazovaného zdroje a žádná data. Protože je použit HTTP protokol verze 1.1, není nutné do požadavku vkládat hlavičku `Connection` pro aktivaci perzistentního spojení – protokol tak činí automaticky.

Požadavek útočníka:

```
HEAD /index.html HTTP/1.1\r\n
Host: [...]\r\n\r\n
```

Odpověď serveru:

```
HTTP/1.1 200 OK\r\n
Date: Mon, 25 Nov 2019 23:10:26 GMT\r\n
Server: Apache/2.4.29 (Ubuntu)\r\n
Last-Modified: Tue, 19 Nov 2019 12:02:59 GMT\r\n
ETag: "2aa6-597b1d868349c"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 10918\r\n
Vary: Accept-Encoding\r\n
Content-Type: text/html\r\n\r\n
```

Jak je vidět z příkladu výše, server odpoví na legitimní požadavek legitimní odpovědí tak, jak se očekává.

Dále je kvůli zachování otevřených perzistentních spojení nutné stanovit maximální možný interval, během kterého se budou pravidelně odesílat udržovací požadavky – pro jednoduchost ty stejné jako úvodní požadavky. Protože je parametr KeepAliveTimeout na serveru ve výchozím stavu nastaven na 5 vteřin, bude vhodné nastavit odesílání udržovacích požadavky s čtyřsekundovými intervaly. Za tuto dobu stihne útočník odeslat validní požadavek, webový server požadavek zpracuje a odešle validní odpověď se všemi náležitostmi, jak je zobrazeno výše. Timeout se tak resetuje, a pokud se některé ze spojení z nějakých důvodů uzavře, například kvůli přesáhnutí maximálního počtu požadavků v rámci jednoho spojení nebo maximální doby jednoho perzistentního spojení, je nahrazeno novým. Celý proces se opakuje v cyklu až do dosažení efektu útoku Slow Next, kdy legitimní uživatel nebude schopen načíst webovou stránku zaneprázdněného webového serveru.

3.2 Bezpečnostní moduly webového serveru Apache

Je také nutno podotknout, že od verze 2.2.15 je na webovém serveru Apache nainstalován modul, který se svou funkcí snaží bránit server před účinky pomalých DoS útoků – `mod_reqtimeout` (dále jako ReqTimeout). Modul poskytuje direktivu, která umožňuje serveru ukončit spojení, pokud zjistí, že klient neposílá data dostatečně rychle [11].

```
RequestReadTimeout header=10-20,MinRate=500 body=20,MinRate=500
```

V tomto příkladu webový server Apache ukončí připojení, pokud klientovi trvá odeslání záhlaví HTTP požadavku déle než 10 sekund. Pokud posílá data záhlaví rychlostí větší než 500 B/s, má na dokončení požadavku déle než 20 sekund. Zároveň ukončí připojení i tehdy, pokud klientovi trvá více než 20 sekund, než zašle tělo požadavku, ale ponechá spojení otevřené tak dlouho, dokud klient odesílá data rychlostí větší než 500 B/s [11].

Tato konfigurace umožňuje klientům se špatným internetovým připojení (jako jsou klienti s vysokou latencí nebo v nízkoúrovňových celulárních nebo satelitních sítích) odesílat požadavky, přičemž stále částečné chrání před pomalými DoS útoky [11]. O účincích na útoky pojednávají kapitoly 8.1.2 a 8.1.3.

Druhým hojně využívaným modulem pro zabezpečení webových serverů proti útokům DoS je modul Mod-Security. Jedná se o otevřený multiplatformní webový aplikační firewall (WAF) vyvinutý společností Trustwave's SpiderLabs. Je velice robustní a poskytuje ochranu před řadou útoků proti webovým aplikacím (SQL injection, cross-site scripting apod.) a umožňuje sledování HTTP provozu, záznam logů a analýzu v reálném čase. Mimo jiné umožňuje ovlivňování maximálního počtu spojení přijatých serverem z jedné IP adresy, což může radikálně ovlivnit účinnost útoků DoS. Tento modul ale není na webovém serveru Apache předinstalovaný [12].

4 Generátor pomalých DoS útoků

V této kapitole je vysvětlen vývoj a fungování generátoru pomalých DoS útoků. Jelikož žádný takový generátor není v současné chvíli dostupný, bylo nutné vytvořit vlastní. Nese název SlowDoSGen. Jeho podrobná konfigurace je popsána v kapitole 8.

Generátor je napsán v programovacím skriptovacím jazyce Python. Jsou k tomu využity tzv. sokety, které propojí počítač útočníka s cíleným serverem. Jsou obsaženy v knihovně `sockets`. Každé perzistentní spojení je tedy reprezentováno tímto soketem, přes který jsou posílány požadavky.

Při spouštění skriptu SlowDoSGen útočník specifikuje základní parametry nutné pro správný běh útoku. Povinnými parametry jsou IP adresa cílené stanice, výběr útoku (Slowcomm, nebo Slow Next), podle čehož se budou posílat vhodné úvodní a udržovací požadavky, počet perzistentních spojení, která má algoritmus navázat s cílenou stanicí a cílový port. Generátor byl vytvořen, optimalizován a testován pro útoky na portech 80, 21 a 22 (HTTP, FTP a SSH). Volitelnými parametry jsou payload, tedy odesílané škodlivé požadavky, které může útočník specifikovat v textovém souboru, dva druhy timeoutů, počet vláken (využívaný pouze při útoku Slow Next). Pokud tyto parametry nejsou útočníkem specifikovány, jsou nastaveny optimální hodnoty získané při testování generátoru (viz kapitola 8). To vše se děje v rámci funkce `main`, jak je vidět na výpisu 4.1.

Generátor z obdržených parametrů rozhodne, jaký útok bude spuštěn a aplikuje definovaný postup. Implementace útoků se mírně liší, zejména v použití rozdílných úvodních a udržovacích požadavků.

Spouští se pomocí příkazové řádky příkazem `python` a jména generátoru s přidruženými parametry (viz obr. 4.1). Pro spuštění je nutná verze `Python 3.0` a vyšší.

Jako dokumentace ke generátoru SlowDoSGen a podrobný popis konfigurace jednotlivých parametrů útoku byl vytvořen i soubor `SlowDoSGen_manual.pdf`, dostupný v příloze B.

Výpis 4.1: Funkce main generátoru SlowDoSGen

```
1 # definice parametrů generátoru
2 ...
3 # inicializace útoku
4 if attack == "C":
5     slowcomm(ip, socket_count, port, timeout, payload)
6 else:
7     start = time.perf_counter()
8     for _ in range(threadCount):
```

```
9           t = threading.Thread(target=slowNext, args=[ip,
                socket_count, port, payload, keep_alive,
                timeout])
10          t.daemon = True
11          t.start()
12          time.sleep(timeout_2)
13      while True:
14          time.sleep(3000)
```

4.1 Popis implementace útoku Slowcomm

Po specifikaci povinných parametrů a výběru útoku Slowcomm je zavolána metoda `slowcomm`. Ta pomocí další funkce `init_socket_C` nejdříve vytvoří určitý počet spojení se serverem podle útočníkem zadaných parametrů a pošle skrze ně úvodní požadavky. Pokud se při navazovaní spojení vyskytne chyba, je generování dalších spojení ukončeno díky definovanému timeoutu pro vytvoření spojení – server už totiž není schopný další spojení navázat a požadavky zahazuje. Ve druhé fázi jsou posílány udržovací požadavky – pro server zdánlivé pokračovaní záhlaví požadavku. Zde se zahajuje nekonečný cyklus. Pokud je spojení i nadále aktivní, pošle se další udržovací požadavek. Pokud bylo spojení z nějakých důvodů ukončeno, je nahrazeno novým. Následně se aplikuje timeout pro co největší zdržení komunikace a donucení serveru čekat na další části požadavku, který se ale nikdy nedokončí. Po jeho uplynutí se cyklus opakuje.

```
[redacted] python SlowDoSGen.py -h
usage: SlowDoSGen.py [-h] -a  -ip  -c  -p  [-l] [-tc] [-t] [-t2]

A tutorial of SlowDoSGen

optional arguments:
  -h, --help            show this help message and exit
  -a , --attack         C = Slowcomm OR N = Slow Next
  -ip , --ipaddress     target IP address or URL
  -c , --connection     number of connections to be established
  -p , --port           destination port
  -l , --payload        load file with custom payload (enter name of file)
  -tc , --threadCount   number of threads to create (for Slow Next only)
  -t , --timeout        timeout sets how often are requests sent in cycle (for both attacks, by Slow Next
max is 5 sec - see persistent connection)
  -t2 , --timeout_2     timeout_2 sets how quickly are the threads started after each other (for Slow Next
 only)
```

Obr. 4.1: SlowDoSGen – generátor pomalých DoS útoků

4.2 Popis implementace útoku Slow Next

Pokud útočník vybere útok Slow Next a zadá povinné parametry, iniciuje se funkce `slownext`, která pomocí další funkce `init_socket_N` navazuje spojení s cíleným serverem a posílá úvodní požadavky. Podobně jako v případě útoku Slowcomm, pouze s tím rozdílem, že funkce `init_socket_N` ověřuje reakci cíleného serveru a čeká na jakoukoliv jeho odpověď. Algoritmus opět ošetřuje výjimku v podobě chyby při navazování TCP spojení. Po přijetí odpovědi serveru se aplikuje timeout pro co největší zpoždění komunikace. Po jeho uplynutí jsou posílány udržovací požadavky v nekonečném cyklu s využitím zmíněného timeoutu. Pokud je zjištěno, že spojení bylo ukončeno, je nahrazeno novým. Celý tento postup je aplikován do vícevláknového systému, kdy každé vlákno vytváří určitý počet spojení. Bylo tak učiněno kvůli výpočetní a časové náročnosti tvorby spojení, kterých je třeba navázat více, než v případě útoku Slowcomm. Server totiž nebyl schopen reagovat na tolik požadavků najednou a zahazoval je. Bylo proto nutné požadavky rozfázovat do několika vln v určitých časových intervalech. Zároveň je toto řešení výhodnější z hlediska eliminace počáteční datové špičky a způsobuje těžší detekci útoku.

5 Obrana proti pomalým DoS útokům

Tato část bakalářské práce se zabývá specifikací a návrhem software, schopného detekovat zkoumané pomalé DoS útoky v síti. Už jen samotné DoS útoky jsou poměrně rychle aktualizovány a využívají stále nové a neznámé zranitelnosti cílených protokolů a systémů. Pomalé DoS se mohou pokládat za další vývojové stádium DoS útoků. Jsou totiž mnohem hůře identifikovatelné v síti. DoS útoky se vyznačují rapidním nárůstem spojení s cíleným serverem během několika málo sekund a jsou jasně rozlišitelné od legitimního síťového provozu – mají za cíl zahltit výpočetní kapacity a nestarají se o nenápadnost celého procesu. Oproti tomu pomalé DoS útoky se zaměřují právě na co největší možné znemožnění identifikace škodlivých spojení pomocí využívání co nejmenší šířky pásma a velikosti posílaných škodlivých dat (viz kapitola 2.1).

V dnešní době jsou navíc nástroje schopné provádět útoky DoS snadno dostupné na Internetu. Je tomu tak již i v případě pomalých DoS útoků. Nástroje jsou propracovány do takové míry, že takto nebezpečné útoky může provést i naprostý laik. U pomalých DoS útoků je navíc schopen ho provést pouze z jednoho počítače, což ještě zjednodušuje celý proces útočení. Pokud se navíc u těchto útoků provede varianta distribuovaného útoku DoS, je detekce ještě složitější. Je proto nutné se ochranou proti podobným útokem zabývat a neustále aktualizovat nové poznatky.

Základní metodou je filtrování IP adres a jejich blokování v případě naplnění určitých podmínek shodujících se s vlastnostmi považovanými za útok. Aktuálně je k dispozici několik metod pro detekci útoků DoS, ovšem ne všechny jsou schopny zachytit i ty nejnovější hrozby. Existuje i možnost manuální kontroly pomocí různých záznamů (logů). Ta se ale v dnešní době zejména kvůli velkému množství dat, které je potřeba prověřit, nepoužívá. Místo této metody nastupují různé bezpečnostní moduly a systémy detekce/prevence průniku [13]. Ke slovu také přicházejí řešení implementovaná například přímo do systému, přidání speciálních filtračních zařízení podobných firewallům do zabezpečované sítě nebo změna celé architektury systému, jako například v případě webového serveru Nginx (viz kap 8.2).

Systémy prevence průniku (IPS – Intrusion Prevention System) a systémy detekce průniku (IDS – Intrusion Detection System) fungují na základě detekce signatur nebo anomálií. Signatura vystihuje určité pravidlo (vzoru útoku), vytvořené v nějakém programovacím jazyce. Pokud některé pakety neprojdou skrze toto pravidlo, je vyhlášena určitá akce, například varování před probíhajícím útokem (IDS) nebo mohou být spuštěny mechanizmy pro zastavení útoku (IPS) [13].

Detekce anomálií se zakládají na vytváření modelů a jejich porovnávání s aktuálním síťovým provozem. Modely se dělí na frekvenční, referenční a model strojového učení. Frekvenční model spočívá v detekci definovaných anomálií. Pokud přesáh-

nou stanovený počet, je vyvolána akce. Referenční model je založen na vytvoření modelu, který stanovuje normální fungování sledovaného systému. Při odchýlení od normálního provozu je vyvolána akce. Model strojového učení využívá algoritmů a statistických metod, které umožňují počítačovému systému imitovat proces učení. Učením je myšlena změna vnitřního stavu systému, která zefektivní schopnost přizpůsobení se změnám okolního prostředí [13].

Tato práce si kromě jiného klade za cíl také vytvoření software pro úspěšnou detekci pomalých DoS útoků Slowcomm a Slow Next, proto se bude zabývat implementací systému detekce průniku IDS.

5.1 Detekce pomalých DoS útoků

Kvůli zmíněným důvodům v úvodu kapitoly 5 se tato kapitola věnuje specifikaci parametrů, které je nutné sledovat pro úspěšnou detekci zkoumaných pomalých DoS útoků – Slowcomm a Slow Next.

Protože pomalé DoS útoky jsou proveditelné i z jednoho počítače, budou v tomto případě obsahovat stejnou zdrojovou IP adresu. Takto vedený útok by bylo možné detekovat stanovením určité hranice počtu spojení na jednoho uživatele – požadavků z jedné IP adresy. Tento postup by ovšem nefungoval proti dnes mnohem více používanějším distribuovaným formám útoků. Je proto vhodné zaměřit se i na jiný postup při detekci útoků.

Pro detekci útoků je možné sledovat určité obecné parametry, které pomalé DoS útoky spolu se známějšími a používanějšími DoS útoky splňují. Tyto parametry jsou nazývány signaturami (známé nepříznivé události). Jsou jimi například:

- *Poměr časového intervalu a počtu právě navázaných spojení:* nejen pomalé DoS útoky se vyznačují zpravidla prudkým nárůstem spojeních navázaných útočníkem, potřebných k zamezení služby. Tento parametr se může sledovat i v případě běžných DoS útoků. Vzhledem k vytíženějším serverům může být ale tento parametr považován pouze za varovný.
- *Velikost přijímaných paketů:* útoky v paketech opakovaně posílají zpravidla stejný objem dat. Legitimní uživatel za normálních okolností takovéto chování nevykazuje, proto se podobný datový tok může považovat za podezřelý.
- *Sledování přítomnosti určitých informací v datovém provozu:* například velké množství opakujících se SYN paketů při útoku TCP SYN flood, nebo třeba v případě útoků Slowcomm a opakujícími se nedokončenými požadavky.

Například celkový objem přenášených dat v síti nemá význam kontrolovat, protože pomalé DoS útoky se snaží imitovat legitimní provoz sítě a jsou tak v tomto ohledu nerozeznatelné – mohlo by se pouze jednat pouze o zvýšený provoz způsobený legitimními uživateli. To se projevuje hlavně v případě útoku Slow Next, kde

nejdůležitější signaturou pro jeho detekci je časový interval mezi požadavky. V případě útoku Slowcomm může podobný provoz generovat legitimní uživatel s pouze pomalým připojením k síti.

Při konkrétní specifikaci signatur pro útoky Slowcomm a Slow Next samozřejmě dochází mezi oběma zkoumanými útoky k rozdílům. Byly zformulovány na základě pozorování chování pomalých DoS útoků generovaných nástrojem SlowDosGen v testovacím prostředí a pomocí zdroje [14].

5.1.1 Signatury pro detekci útoku Slowcomm

Pro detekci útoků typu Slowcomm je nutné sledovat několik parametrů probíhající komunikace s chráněným serverem. Jelikož je útok Slowcomm založen na navázání několika spojení a zasílání nevalidních paketů (viz kapitola 2.1.1), za důležité signatury byly zvoleny následující:

- *Sledování neuzavřených paketů*: v případě protokolu HTTP absence znaků dvojitého odřádkování `\r\n\r\n`.
- *Počet aktivních spojení navázaných z jedné zdrojové IP adresy:* pomalé DoS útoky nevyžadují k úspěšnému útoku velkou výpočetní kapacitu, a proto je možné je provádět i z jednoho počítače. V případě dnes mnohem využívanější distribuované varianty útoku DoS by ale tento parametr nemusel přinést vypovídající hodnoty. Hraniční je hodnota 20 spojení z jedné IP adresy (klasický uživatel si neotevře více než 20 aktivních oken).
- *Počet požadavků v rámci jednoho spojení a doba jejich trvání:* zde se monitoruje maximální počet nevalidních požadavků v rámci jednoho spojení z jedné IP adresy. Kontroluje se i čas, po který jsou již aktivní. Hraniční hodnota je stanovena na 10 požadavků do 15 sekund.

5.1.2 Signatury pro detekci útoku Slow Next

Pro detekci útoků typu Slow Next, který posílá validní požadavky (viz kapitola 2.1.2), je nutné sledovat následující signatury:

- *Sledování uzavřených paketů:* díky tomu, že se požadavky útočníka validní, je tento útok mnohem nebezpečnější a mnohem těžší pro detekci. Je nutné odposlouchávat mnohem širší spektrum síťového provozu. Požadavky v případě protokolu HTTP musí obsahovat znaky `\r\n\r\n`.
- *Časový interval mezi odpovědí a příjmem paketů v rámci jednoho spojení:* útok využívá jistých definovaných timeoutů pro zasílání paketů v cyklu (viz kapitola 3.1). Pokud je tedy zaznamenána určitá strojová přesnost intervalu příjmu nových paketů, které by člověk nebyl schopný generovat, vzniká podezření na

probíhající útok. Hraniční hodnota je stanovena na časový interval mezi požadavky větší než 2 sekundy s tím, že rozdíl mezi hodnotami časového odstupu mezi nimi nesmí překročit 0.1 sekundy.

- *Stále se opakující legitimní dotaz na jeden a ten samý zdroj:* například HTML kód webové stránky nebo obrázek na ní umístěný. Pokud se útočník z hlediska jednodušší implementace nedotazuje na různé zdroje, může být takto odchycen pomalý útok DoS. Tato kontrola probíhá pomocí monitorování velikosti přijímaných požadavku. Hraniční hodnota je stanovena na maximální rozdíl 10 bajtů.
- *Počet aktivních spojení navázaných z jedné zdrojové IP adresy:* v tomto případě musí být kladen důraz na určité zpoždění při průběhu detekčního systému. Pro úspěšnou identifikaci útoku je potřeba nasbírat více dat. Hraniční hodnota signatury je nastavena na maximálně 20 spojení z jedné IP adresy.

Detekce útoku Slow Next se může pokládat za obtížnější v porovnání s detekcí útoku Slowcomm. Protože nevykazuje na první pohled známky škodlivých požadavků, je nutné dbát větší opatrnosti při rozhodnutích, zda se jedná o útok či nikoliv – hrozí totiž záměna s požadavky legitimního uživatele, která je nepřípustná. Pokud jsou signatury například při distribuované verzi tohoto útoku nějakým způsobem obejity, např. střídáním požadavků a mírným zpožděním při jejich zasílání, nelze pomocí tohoto systému IDS útoku zabránit.

6 Implementace vlastního IDS

Tato kapitola se zabývá popisem implementace vlastního systému detekce průniku IDS zkoumaných pomalých DoS útoků typu Slow Next a Slowcomm, který byl vybrán z výčtu možných metod detekce zmíněných v kapitole 5. Systém byl vytvořen na základě definovaných signatur, popsaných v kapitole 5.1 a je schopný detekovat útoky na protokolech aplikační vrstvy HTTP, FTP a SSH. Tzn. těch protokolů, které byly zkoumány při vývoji generátoru pomalých DoS útoků SlowDoSGen.

Některé signatury jsou natolik specifické, že jsou sami o sobě schopné identifikovat probíhající útok. Některé z nich naopak slouží k dosažení větší pravděpodobnosti správnosti detekce útoku. Bylo proto nutné jednotlivá pravidla sestavit do vhodné posloupnosti, kde v případě nedostatečné pravděpodobnosti určení útoku se pomocí dalších signatur útok detekuje jednoznačně.

Výsledný systém je určen pro instalaci přímo na chráněný server s operačním systémem Linux (testován na distribuci Ubuntu). Systém je implementován jako konzolová aplikace, vytvořená v programovacím jazyce Python. Bylo zde využito knihovny `Scapy`, která nativně funguje na všech distribucích operačního systému Linux a Unix s podporou knihovny LibPcap. Dohromady tak slouží k odposlechu síťového provozu na daném rozhraní. S určitým přizpůsobením funguje i na operačním systému Windows.

Detekční systém postupně filtruje potřebná data z podezřelého síťového provozu, která ukládá do databáze podezřelých spojení. Pro konstrukci databáze byla v jazyce Python využita kolekce `dictionary`, která obsahuje dvojice klíčů a hodnot (více v kapitole 6.1). Vzhledem k podobnosti obou zkoumaných pomalých DoS útoků je jedna databáze plně dostačující. Při každém zachycení dalšího paketu se provádí vyhodnocení získaných dat. Pokud některé spojení vykazuje hodnoty definované signaturami (viz kapitola 5.1) přesahující stanovenou hranici, je vyhlášeno varování v podobě probíhajícího útoku zobrazeného v konzoly. Zároveň jsou ukládány informace, na základě kterých byla situace vyhodnocena jako útok, do log souboru. Zdrojová IP adresa útoku je potom odstraněna z databáze, zapsána do tzv. blacklistu (soubor s IP adresami, které mají zakázanou komunikaci se serverem) a systém pokračuje ve sledování síťového provozu s tím, že vyřazenou IP adresu ignoruje, aby byl proces detekce dále přehledný. Seznam blacklist může být následně využit při implementaci systému IPS, kdy při dalším útočníkovu pokusu bude spojení s touto zdrojovou IP adresou rovnou blokováno.

Protože se útoky snaží imitovat legitimní síťový provoz, je nutné nejdříve nasbírat větší množství dat a až poté začít s jejich vyhodnocováním. V případě útoku Slowcomm se totiž může pouze jednat o legitimního uživatele s pomalým připojením k síti, a nebo v případě útoku Slow Next o legitimního uživatele zobrazujícího si

stejný obsah (webovou stránku) ve více oknech prohlížeče (pokud mluvíme o útoku na webový server).

Pro každý útok byly tyto hraniční parametry zvoleny zvlášť, aby bylo dosaženo větší přesnosti a spolehlivosti detekčního systému IDS. Hodnoty byly vytvořeny způsobem definování takových vlastností síťového provozu, kterých legitimní uživatel nemůže dosáhnout, ale zkoumané DoS útoky se takovým chováním vyznačují. Maximální počet spojení z jedné IP adresy (parametry `SC_max_conn_from_IP` a `SN_max_conn_from_IP`) byl stanoven na 20, kdy průměrný uživatel zpravidla neotevře více než tento počet aktivních oken nebo spojení [14]. Maximální počet požadavků v rámci jednoho spojení (`SC_max_conn_req_from_IP`) byl v případě útoku Slowcomm stanoven na maximálně 20 neukončených požadavků. Tato hodnota byla vytvořena s určitou rezervou, kdy legitimní provoz jednoduše nemůže přesáhnout tuto hodnotu. Pro útok Slow Next nemůže být tento parametr brán v potaz, protože se jedná o zcela legitimní požadavky a z toho důvodu nemohou být nijak omezovány. Dalším je parametr maximálního časového intervalu od začátku monitorování podezřelého spojení (`SC_max_from_start_time`). Pokud je již přijato alespoň 10 požadavků (parametr `SC_max_conn_req_from_IP_time_test`) a všechno by probíhalo legitimně, tento proces by neměl přesáhnout stanovenou hodnotu 10 sekund. Neukončené požadavky se totiž zpravidla vážou k získání informací z určitého zdroje (např. stahování obrázku), který se nemusí vejít do jednoho požadavku. Podle specifických podmínek útoku Slow Next na webový server byla hodnota signatury časový interval mezi odpovědí a příjmem paketů v rámci jednoho spojení nastavena pomocí dvou parametrů na minimálně dvousekundový odstup mezi požadavky (`SN_min_time_interval`) s tím, že rozdíl mezi hodnotami časového odstupu nesmí být větší než 0.1 sekundy (`SN_max_time_difference`) – tento parametr slouží pro menší odchylky způsobené přenosem dat a jejich zpracováním. Rozdíl minimálně 2 sekund odlišuje legitimní uživatele od útočníkem generovaných požadavků, kteří všechna data (např. webové stránky) obdrží zhruba do jedné vteřiny, tudíž je tento rozdíl v reálném provozu nemožný. Parametr `SN_max_size_difference` stanovuje maximální rozdíl velikostí požadavků, který, pokud je splněn, vystihuje chování útoku Slow Next. Má nastavenou hodnotu tolerance rozdílu o 10 znaků. Posledním parametr (`Max_nonactive_time`) označuje dobu, po které je neaktivní spojení odstraněno z databáze. Pomocí těchto parametrů dochází k jednoznačné identifikaci zkoumaných útoků (viz výpis 6.1).

Výpis 6.1: Hraniční parametry detekčního systému

```
1    # Slowcomm parameters
2    SC_max_conn_from_IP = 20
3    SC_max_conn_req_from_IP = 20
```

```
4     SC_max_conn_req_from_IP_time_test = 10
5     SC_max_from_start_time = 15.0
6
7     # Slow Next parameters
8     SN_max_conn_from_IP = 20
9     SN_max_conn_req_from_IP = 25
10    SN_max_time_difference = 0.1
11    SN_max_time_interval = 2.0
12    SN_max_size_difference = 10
13
14    Max_nonactive_time = 20.0
```

6.1 Průběh zachycení a vyhodnocení dat

Jak bylo již popsáno v kapitole 6, pakety jsou zachytávány díky knihovně `Scapy`. Při jejím použití je stěžejním krokem inicializace funkce `sniff`, která je vidět na výpisu 6.2. Parametry této funkce jsou zvolené rozhraní (`iface`), na jakém má probíhat odposlech síťového provozu, dále spuštění funkce, která se vykoná při každém příchodu nového paketu (`prn`) a filtr příchozího provozu (`filter`). Díky němu je odposloucháván provoz pouze z určitých TCP portů a pouze příchozí pakety tím, že pakety musí mít cílovou adresu chráněného severu. Jinak by byly odposlouchávány i odchozí pakety, které ale pro tento účel není nutné zpracovávat. Funkce se nativně spouští v novém samostatném procesu.

Funkce `ids` v parametru `prn` se stará o extrakci informací z příchozích paketů, jako je IP adresa, příchozí a odchozí port, velikost paketu a informace, zda je paket ukončený nebo ne (jestli obsahují ukončovací znak `\r\n\r\n` v případě útoku na protokol HTTP a znak `\r\n` v případě útoku na protokoly FTP a SSH).

Výpis 6.2: Funkce sniff detekčního systému

```
1     sniff(iface='enp0s3', prn=ids, filter="tcp dst port
        21 or 22 or 80 and dst host 10.10.0.2")
```

Pomocí zjištěných parametrů následuje volání funkce pro zápis dat do databáze (její struktura je zobrazeny ve výpisu 6.3) – funkce `isEndedCheck_and_store`. Tato funkce zároveň kontroluje, zda po nedokončeném požadavku nepřišel požadavek ukončený, případně naopak. To by indikovalo, že ani jeden ze zkoumaných útoků neprobíhá a spojení by bylo odstraněno z databáze podezřelých spojení. Tato funkce rovněž kontroluje, jestli nejsou spojení příliš dlouho neaktivní. Pokud jsou neaktivní déle než 20 sekund, odstraňuje příslušnou IP adresu z databáze.

V případě, že funkce provede nový zápis do databáze, je spuštěna buďto funkce `IDS_Slowcomm`, nebo `IDS_SlowNext` podle toho, zda zkoumané spojení obsahuje pouze neukončené, nebo ukončené požadavky. Podle parametrů spojení z databáze se provádí kontrola, jestli neprobíhá útok, tzn. hledají se v databázi spojení, do kterého patří právě příchozí paket a v něm signatury útoku, popsané v kapitole 5.1. Následně se prověří, jestli některá signatura nepřekročila definovanou hranici. V případě překročení této hranice je vyhlášeno varování v podobě probíhajícího útoku zobrazeného v konzoly. Zároveň jsou ukládány informace, na základě kterých byla situace vyhodnocena jako útok do log souboru.

Výpis 6.3: Struktura databáze detekčního systému

```
ids_dict = {IP adresa: [{zdrojový port : [[příchozí
   časy paketů], [velikosti přijatých paketů],
   identifikátor ne/ukončeného paketu, cílový port
   ]}]}
```

Podrobný průběh systému detekce průniku ilustrují vývojové diagramy v přílohách A.1 pro útok Slowcomm a A.2 pro útok Slow Next.

6.2 Spuštění systému detekce průniku

Pro správné fungování detektoru je nutná instalace zmíněné knihovny `Scapy` pomocí příkazu `sudo pip3 install scapy`. Program je také nutné spouštět s administrátorskými právy pomocí příkazové řádky příkazem `sudo python3 SlowDoS_IDS.py`. Pro spuštění je nutná verze `Python 3.0` a vyšší.

Program vyzve uživatele k výběru jednoho z dostupných rozhraní na chráněném serveru. Dále získá jeho IP adresu a uloží ji do zmíněného filtru funkce `sniff`. Dále už pracuje automaticky, pouze v případě detekce útoku zobrazuje hlášení.

6.3 Detekce distribuované formy útoků DoS

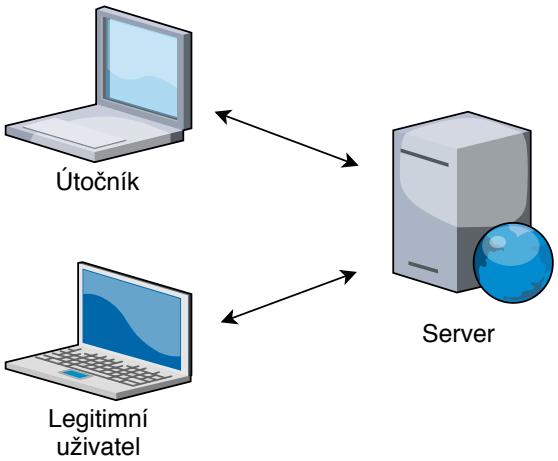
Proces detekce zkoumaných útoků v distribuované formě probíhá naprosto stejně, i kdyby útočník použil stovky nebo tisíce botů v botnetu (viz kapitola 2) a z každého vytvářel pouze jedno spojení na cílený server, čímž by se snažil vyhnout definovaným hraničním hodnotám signatur útoku. K detekování útoku pouze nedojde z důvodu velkého počtu spojení, ale kvůli jiným detekovaným signaturám posílaných paketů. Jak bylo popsáno v kapitole 5.1, například díky monitorování počtu neukončených požadavků, nebo jejich přijímání v pravidelných intervalech. Simulace této formy útoku jsou provedeny v kapitole 9.1.

7 Testovací prostředí

Tato kapitola se zabývá topologií testovacího prostředí (viz obr. 7.1), využitými komponenty a jejich nastavením.

Testovací prostředí je realizováno v multiplatformním virtualizačním open-source nástroji Virtualbox na hostujícím operačním systému Windows 10 Home 64-bit. Pro naplnění zadání bakalářské práce je potřeba vytvořit několik virtuálních strojů. Stroj útočníka, legitimního uživatele a server pro služby HTTP a další zkoumané protokoly. Zároveň budou všechny virtuální stroje umístěny do interní sítě, aby nemohly nijak ovlivnit chod hostujícího systému a případně síť, do které je hostující počítač připojen.

Pro samotné testování bude využit protokolový síťový analyzér `Wireshark` instalovaný přímo na serveru, aby bylo dosaženo maximální spolehlivosti a nenarušení datového přenosu. Pomocí něj je možné sledovat veškerý provoz v interní síti. Konkrétně komunikaci mezi útočníkem, legitimním uživatelem a serverem. Dále nástroj `tcptrack` pro monitorování navázaných, případně uzavíraných spojení, opět přímo na serveru.



Obr. 7.1: Topologie testovacího prostředí

V principu útočník spustí příslušný generátor pomalých DoS útoků například na cílený webový server. Pokud bude dosaženo vyčerpání zdrojů serveru, dodatečný pokus od legitimního uživatele připojit se na webovou stránku provozovanou webovým serverem bude neúspěšný. Bylo dosaženo pomalého útoku DoS.

7.1 Interní síť

V rámci vytvoření bezpečných podmínek pro testování pomalých DoS útoků byla vytvořena interní síť v aplikaci Virtualbox. To znamená, že virtuální stroje nemají přístup k internetu, a ani nemohou komunikovat s hostujícím počítačem. Všem virtuálním strojům byly nastaveny statické IP adresy v síti 10.10.0.0/24. Nastavení statické adresy v případě Kali Linux probíhá klasicky v konfiguračním souboru

`/etc/network/interfaces`.

Na operačním systému Ubuntu od verze 18.04 LTS už ale neprobíhá konfigurace IP adres v tomto souboru, ale defaultně využívá utilitu Netplan a její konfigurační YAML soubor

`/etc/netplan/01-netcfg.yaml`.

V tomto souboru jsou definována rozhraní, jejich IP adresy, využití DHCP serveru apod. Z tohoto souboru poté Netplan při startu systému vygeneruje veškerou potřebnou konfiguraci [15].

7.2 Využité servery

Pro testování je využit v dnešní době nejvyužívanější webový server pro hostování webových stránek – Apache verze 2.4.29, ponechán ve výchozím nastavení s použitou testovací webovou stránkou [16]. Protože se tato práce zabývá DoS útoky, není zapotřebí žádných úprav webových stránek. Dále FTP server vsftpd verze 3.0.3 a nástroj OpenSSH pro testování protokolu SSH. Pro další testování byly nainstalovány i webové servery Nginx, lighttpd a IIS od Microsoftu. Všechny kromě serveru IIS jsou nainstalovány na stanici s linuxovým operačním systémem Ubuntu. Webový server IIS je nainstalován ve virtuálním stroji Windows Server 2019 Standard.

Parametry serveru Ubuntu:
Hostitelský operační systém: Microsoft Windows 10 Home 64-bit
Operační systém: Ubuntu 18.04.3 LTS 64-bit
Procesor: Intel® Core™ i7-8750H (v rámci Virtualboxu povolena 2 jádra)
RAM: 1024 MB
IP adresa: 10.10.0.2
Další software: Wireshark, tcprack, OpenSSH, servery Apache, vsftpd, lighttpd a Nginx

Parametry serveru IIS:
Hostitelský operační systém: Microsoft Windows 10 Home 64-bit
Operační systém: Windows 2016 64-bit
Procesor: Intel® Core™ i7-8750H (v rámci Virtualboxu povolena 2 jádra)
RAM: 1024 MB
IP adresa: 10.10.0.3

7.3 Útočník

Jako útočník je v této topologii využit virtuální stroj s operačním systémem Kali Linux. Jedná se o linuxovou distribuci odvozenou od operačního systému Debian, navrženou pro digitální forenzní analýzu a penetrační testy. Z tohoto virtuálního stroje se tedy vedou útoky na server pomocí generátoru pomalých DoS útoků.

Parametry stroje útočníka:
Hostitelský operační systém: Microsoft Windows 10 Home 64-bit
Operační systém: Kali Linux
Procesor: Intel® Core™ i7-8750H (v rámci Virtualboxu povolena 2 jádra)
RAM: 3021 MB
IP adresa: 10.10.0.4
Další software: Wireshark 2.6.10-1

7.4 Legitimní uživatel

Jako oběť nepřístupnosti serveru je v této topologii zvolen virtuální stroj s operačním systémem Windows 10 64-bit. Jediný úkol tohoto uživatele je opakovaně se pokoušet o připojení k serveru a testovat jeho dostupnost. V případě, že byl útok DoS proveden správně, tak se legitimnímu uživateli připojit nepodaří.

Parametry stroje legitimního uživatele:
Hostitelský operační systém: Microsoft Windows 10 Home 64-bit
Operační systém: Microsoft Windows 10 Home 64-bit
Procesor: Intel® Core™ i7-8750H (v rámci Virtualboxu povoleno 1 jádro)
RAM: 2 GB
IP adresa: 10.10.0.5

8 Testování pomalých DoS útoků

Tato kapitola se zabývá výsledky testování, analýzou a účinností pomalých DoS útoků generovaných nástrojem SlowDoSGen na webový server Apache, FTP server vsftpd a službu OpenSSH.

Průběh útoku je vždy znázorněn grafem, kde oranžová křivka popisuje spojení navázaná útočníkem a modrá dostupnost serveru z hlediska potenciálně volných spojení pro legitimní uživatele.

8.1 Testování webového serveru Apache

V této kapitole jsou popsány výsledky testování webového serveru Apache pomocí generátoru SlowDosGen a jeho výchozí nastavení. Zároveň byl separátně testován bezpečnostní aktivní a vypnutý modul ReqTimeout spolu s modulem Mod-Security (viz kapitola 3.2). Modul Mod-Security ovšem ve výchozím nastavení neměl na průběh zkoumaných útoků Slowcomm a Slow Next žádný účinek, a proto není dále nijak zkoumán.

8.1.1 Nastavení webového serveru Apache

V rámci testování byla konfigurace webového serveru Apache ponechána ve výchozím nastavení. Jak již bylo zmíněno v kapitole 3.1, pro testování je prospěšné vědět, jaká tato nastavení jsou [17].

Maximální počet perzistentních spojení je nastaven na 150. To je určeno nastavením v souboru `/etc/apache2/mods_available/mpm_prefork.conf` hodnotou `MaxRequestWorkers`, jak je vidět ve výpisu 8.1.

Výpis 8.1: Nastavení modulu mpm_prefork

```
1     <IfModule mpm_prefork_module>
2         StartServers                5
3         MinSpareServers             5
4         MaxSpareServers             10
5         MaxRequestWorkers           150
6         MaxConnectionsPerChild      0
7     </IfModule>
```

Dalšími důležitými hodnotami pro útoky jsou jednotlivé timeouty a nastavení perzistentních spojení Keep–Alive (viz kapitola 3.1). Ty najdeme definované v souboru `/etc/apache2/apache2.conf`, viz výpis 8.2.

Již zmíněný modul ReqTimeout (viz výpis 8.3) webového serveru Apache je ve výchozím stavu zapnutý. Jeho nastavení je popsáno v konfiguračním souboru `/etc/apache2/mods-available/reqtimeout.conf` (viz kapitola 3.2). Toto je jediné nastavení, se kterým se při testování manipulovalo.

Výpis 8.2: Konfigurační soubor apache2.conf

```
1    Timeout                 300
2    KeepAlive               On
3    MaxKeepAliveRequests    100
4    KeepAliveTimeout        5
```

Výpis 8.3: Konfigurační soubor reqtimeout.conf

```
1    <IfModule reqtimeout_module>
2        RequestReadTimeout header=20-40,minrate=500
3        RequestReadTimeout body=10,minrate=500
4    </IfModule>
```

8.1.2 Slowcomm

V první části je probrán test s vypnutým bezpečnostním modulem ReqTimeout. Účinnost útoku na webový server Apache se zapnutým modulem ReqTimeout je popsána v druhé části této kapitoly.

Útok je spouštěn v následující konfiguraci. Otevírá 225 spojení s webovým serverem, tj. více než je maximální počet, který je server schopen navázat. Tato optimální hodnota byla zjištěna testováním. Samozřejmě lze zvolit jakoukoliv vyšší hodnotu. Timeout pro posílání udržovacích požadavků je stanoven na 7 sekund. Port je nastaven na hodnotu 80 – službu HTTP. IP adresa, na které se nachází virtuální stroj s operačním systémem Ubuntu a samotný cílený webový server Apache, je 10.10.0.2.

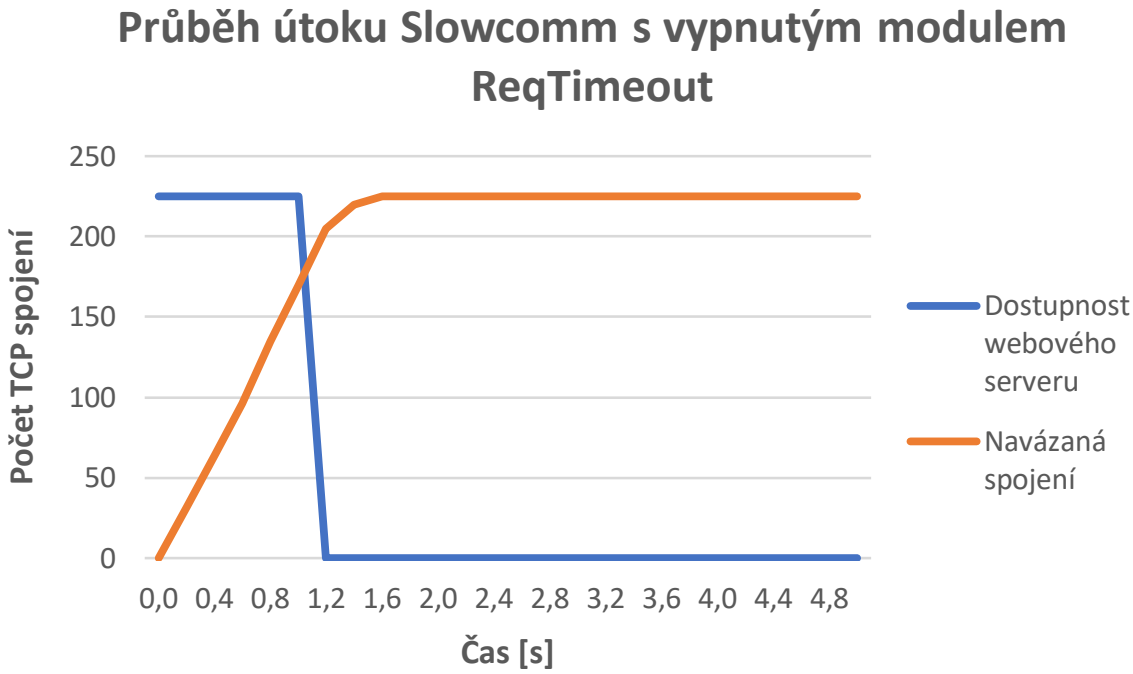
Příkaz pro spuštění útoku v nástroji SlowDoSGen v příkazové řádce:

```
python3 SlowDoSGen.py -a C -ip 10.10.0.2 -c 225 -p 80
```

Vypnutý modul ReqTimeout

Bez použití ochran webového serveru proti DoS útokům a se zvolenou konfigurací bylo dosaženo efektu pomalého DoS útoku téměř okamžitě, ihned po odeslání úvodních nedokončených požadavků (1,23 sekund) a trvá nepřetržitě. Server stále čeká na doručení zbytku požadavku, přičemž tím blokuje ostatní legitimní uživatele k navázání spojení, jak je vidět na obrázku 8.1, kdy útočník naváže TCP spojení (oranžová křivka) a server již není schopen poskytnout další spojení legitimním uživatelům

(modrá křivka). Spojení uzavře až timeout pro maximální délku perzistentního spojení, který je 300 sekund (viz kapitola 8.1.1). Následně se ale díky implementaci útoku spojení obnoví.



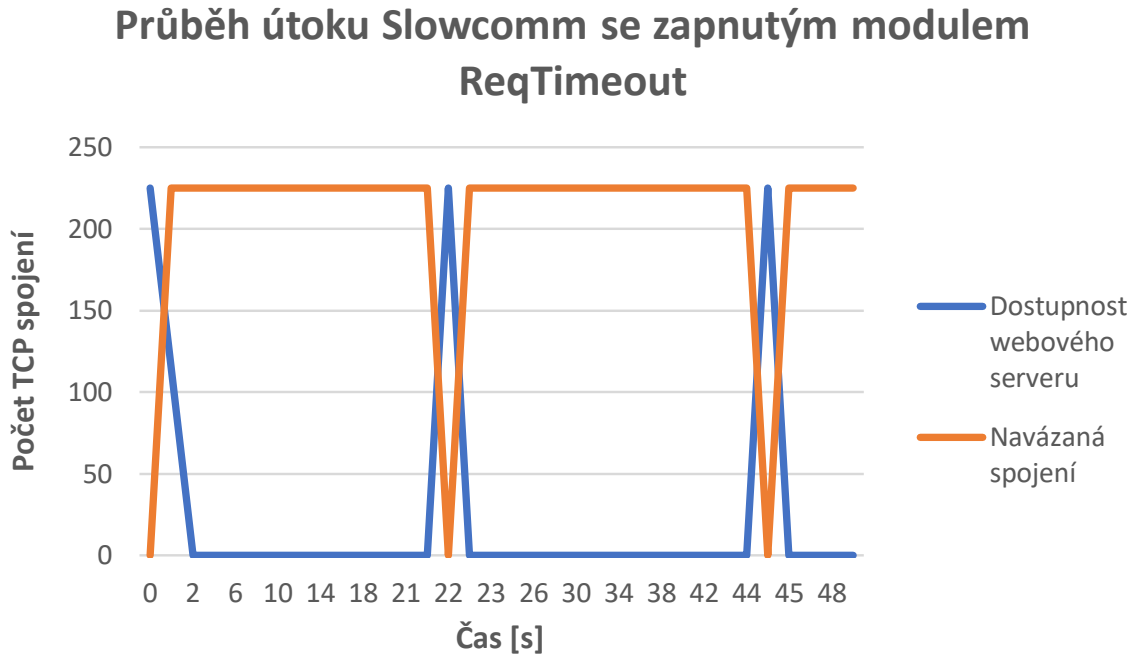
Obr. 8.1: Průběh útoku Slowcomm s vypnutým modulem ReqTimeout

Bylo také zjištěno, že při takto nakonfigurovaném serveru není nutné odesílat udržovací požadavky. Webový server má v tomto případě pouze dvě omezení, která by způsobovala uzavření sestavených spojení mezi útočníkem a serverem. Prvním je již zmíněný timeout 300 sekund. Jedná se o poměrně dlouhý interval, v rámci kterého by legitimní uživatel jistě webovou stránku opustil. Proto by timeout mezi odesíláním udržovacích požadavků mohl být nastaven na 302 sekund. Dvě sekundy by byly ponechány jako rezerva pro ukončení všech spojení. Druhým omezením je maximální počet odeslaných požadavků v rámci jednoho spojení, tj. 100. Po těchto časových intervalech je ale spojení opět obnoveno pomocí udržovacích požadavků. Útok je potom i nadále úspěšný.

Aktivní modul ReqTimeout

V případě zapnutého bezpečnostního modulu se aplikuje pravidlo zmíněné v kapitole 3.2. Tím je zajištěno, že server ukončí spojení po 20 až 40 sekundách v případě, že je záhlaví požadavku posíláno nižší rychlostí než 500 B/s, nebo po 10 sekundách, pokud je tělo požadavku odesíláno rychlostí menší než 500 B/s. To odpovídá parametrům provozu útoku Slowcomm.

Toto nastavení se ale jednoduše obejde buďto obnovováním spojení každých 22 vteřin (rezerva 2 sekundy pro samotný proces obnovování spojení), kdy server zhruba po 20 sekundách spojení ukončí, ale algoritmus spojení ihned obnoví, nebo naopak častějším obnovováním požadavků například každé 3 sekundy, které zajistí větší stabilitu útoku z hlediska navázaných spojení útočníkem.



Obr. 8.2: Průběh útoku Slowcomm se zapnutým modulem ReqTimeout

Jak je znázorněno na obrázku 8.2, každých zhruba 20 vteřin server ukončí všechna spojení, algoritmus útoku je ale hned začne obnovovat. V této chvíli se legitimní uživatel může stihnout připojit, ovšem zhruba po 1 sekundě znovu dojde k zamezení služby. Útok Slowcomm byl tedy opět úspěšný a znepřístupnil stránku legitimnímu uživateli.

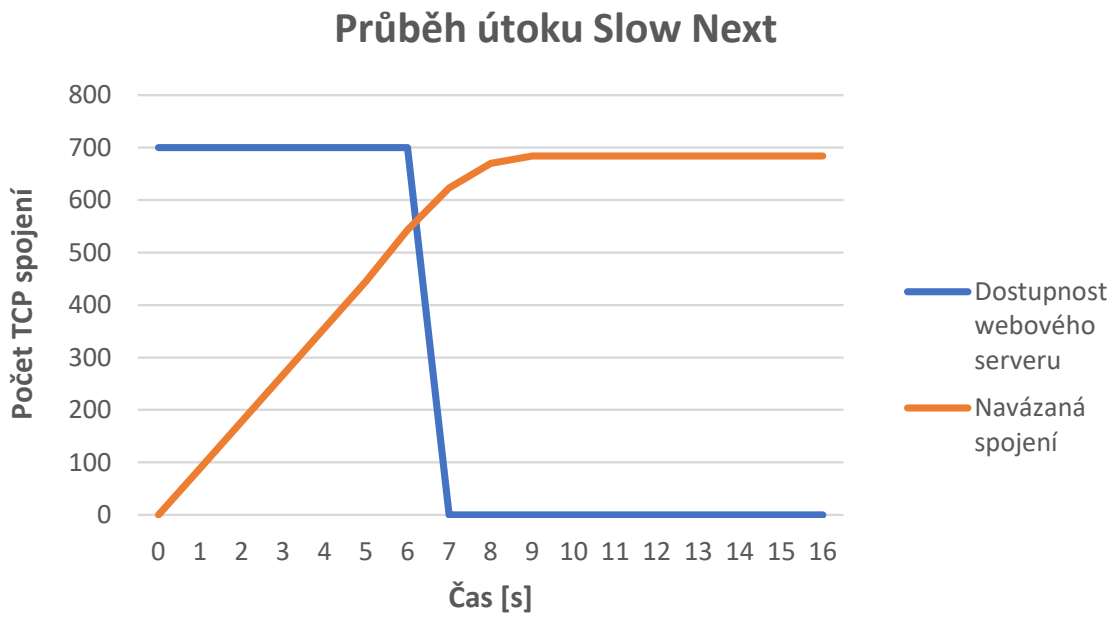
8.1.3 Slow Next

Útok je spouštěn v následující konfiguraci. Otevírá 700 spojení (5 vláken – výchozí nastavení generátoru, které se nemusí uvádět v příkazu, každé otevírá 140 spojení) s webovým serverem na IP adrese 10.10.0.2. Timeout pro posílání udržovacích požadavků je stanoven na 4 sekundy. Port je nastaven na službu HTTP, tedy 80.

Příkaz pro spuštění útoku v nástroji SlowDoSGen v příkazové řádce:

```
python3 SlowDoSGen.py -a N -ip 10.10.0.2 -c 140 -p 80
```

Testování proběhlo podobně jako v případě útoku Slowcomm. Nejprve s vypnutým, a potom se zapnutým modulem ReqTimeout. Protože jsou ale v tomto útoku využívány legitimní požadavky, ani bezpečnostní modul ReqTimeout nevyhodnotí tuto komunikaci za škodlivou, což je potvrzeno následujícím testováním. Jak můžeme na obrázku 8.3 pozorovat, navázání spojení trvá zhruba 6 sekund. Webový server ale není schopen takové množství spojení udržet. Vytvoří pouze 684 spojení a ostatní z důvodu jeho vyčerpané kapacity zahazuje. Proto ihned po navázání spojení dochází k zamezení služeb legitimnímu uživateli, který webovou stránku nenačte.



Obr. 8.3: Průběh útoku Slow Next

Rozdíl oproti útoku Slowcomm můžeme pozorovat v přenosových rychlostech. Útok Slow Next je několikanásobně náročnější a posílá data průměrnou rychlostí 20 125 B/s. To je dáno tím, že jsou zde posílány legitimní požadavky, takže dojde vícekrát k posílání celého požadavku a hlavně celé odpovědi serveru. Stále se ale tato hodnota může považovat za nízkou a nijak zvlášť podstatnou při detekci útoku. Druhý rozdíl je v počtu nutných navázaných spojení pro dosažení útoku DoS. Minimální počet spojení je okolo 650, což je výrazně více, než při útoku Slowcomm. To může být způsobeno pro server menší náročností zpracovávat kompletní legitimní požadavky, než vyčleňováním paměti pro čekání na dokončení požadavku v případě útoku Slowcomm.

Útok Slow Next je tak úspěšnější než útok Slowcomm. Vyvolal zamezení služby v obou scénářích testování beze změn v konfiguraci.

8.2 Další webové servery

Protože DoS útoky se nejčastěji používají na protokol HTTP, nabízí se možnost vyzkoušet funkčnost útoků i na některé další webové servery. Tato kapitola je zaměřena na otestování dalších HTTP serverů, které paří mezi ty nejrozšířenější – Internet Information Server verze 10.0 (dále jen IIS) od firmy Microsoft, Nginx verze 1.14.0, lighttpd verze 1.4.45. Tabulka 8.1 znázorňuje, zda byly útoky úspěšné či nikoli.

Tab. 8.1: Přehled účinnosti generátoru SlowDoSGen na webových serverech

Webový server	Slowcomm	Slow Next
Apache	✓	✓
Nginx	✓	×
IIS	×	×
lighttpd	✓	✓

Útoky Slowcomm a Slow Next fungují na principu obsazování maximálního počtu spojení a tím vyčerpání výpočetní kapacity serveru (viz kapitoly 2.1.1 a 2.1.2). V případě dříve testovaného webového serveru Apache byly útoky úspěšné, protože je tento server založen na vícevláknovém zpracovávání jednotlivých spojení – server pro každé spojení vytvoří jedno vlákno. Nárůst spojení (tzn. vláken či procesů) způsobený útoky zapříčiní vyčerpání kapacity serveru [18].

Oproti tomu, server Nginx má rozdílnou architekturu řízenou událostmi, tzn., že tento webový server dokáže obsloužit mnohonásobně více požadavků najednou s minimálními nároky na operační paměť [19]. To způsobí, že by měl být imunní vůči útokům generátoru SlowDoSGen. Při útoku Slow Next se toto opravdu projevilo a útok nebyl úspěšný. I při útoku Slowcomm byl server mnohem odolnější než webový server Apache, ovšem po navázání 800 spojení došlo ze strany serveru k mohutnému ukončování spojení, což paradoxně vytížilo server natolik, že to způsobilo nedostupnost webové stránky a efekt útoku DoS.

Další dva zkoumané webové servery splnily očekávání. Server IIS, který využívá asynchronně zpracovávané skupiny vláken (tj. spojení s klienty) neboli tzv. pool, byl naprosto imunní vůči všem konfiguracím útoků [20]. Oproti tomu server lighttpd, který byl vytvořen tak, aby byl rychlý a hlavně velmi málo náročný na operační paměť, podlehl oběma pomalým DoS útokům [21]. Pouze vzhledem k jeho nenáročnosti na výpočetní kapacity bylo nutné navázat více spojení pro vyčerpání kapacity serveru – konkrétně 400 spojení v případě obou útoků.

8.3 Testování na protokolu FTP

Testování bylo dále provedeno na protokolu aplikační vrstvy FTP, nainstalovaném na unixovém FTP serveru vsftpd. Test za použití stejných payloadů byl úspěšný pouze v případě útoku Slowcomm. Tímto bylo potvrzeno, že při útoku Slowcomm na jakýkoliv protokol nezáleží na použitém payloadu. Útok Slow Next nebyl úspěšný, tzn. nedošlo k zamezení služby. Byla proto nutná analýza protokolu (viz kapitola 1.2), na jejímž základě byl definován nový payload (úvodní požadavek) pro útok na protokol FTP. Bylo využito příkazu USER, k němuž jsou náhodně generována uživatelská jména o délce 5 znaků.

```
USER <jméno>\r\n
```

Je také důležité zmínit, že jak protokol HTTP, i protokol FTP má v konfiguračních souborech definovaná určitá omezení. Nemá sice žádný timeout na spojení, ale obsahuje omezení v podobě maximálního počtu spojení z jedné IP adresy, kterých je 50. Tento počet spojení nedokázal způsobit efekt DoS útoku, protože FTP server je schopen navázat až 500 spojení s různými uživateli zároveň. Z těchto důvodů bylo nutné toto omezení vypnout, aby bylo možné provést testování [9]. Toto nastavení by se jinak mohlo obejít například použitím distribuované verze útoků.

Testováním bylo zjištěno, že pokud je útočník schopen navázat 2000 a více spojení s FTP serverem, docílí se zamezení služby a útoku DoS téměř okamžitě, jak je vidět na obrázku 8.4. Útočník navázal 2000 spojení se serverem (oranžová křivka), který tak není schopný navázat další spojení s legitimními uživateli (modrá křivka).

V případě použití útoku **Slowcomm** byl využit výše definovaný payload (úvodní požadavek) `USER <jmeno>` bez ukončovacích znaků `\r\n`. Jako udržovací paket je zde opět použito písmeno „i“. Tím je docíleno toho, že FTP server stále čeká na dokončení uživatelského jména, čímž blokuje připojení legitimních uživatelů.

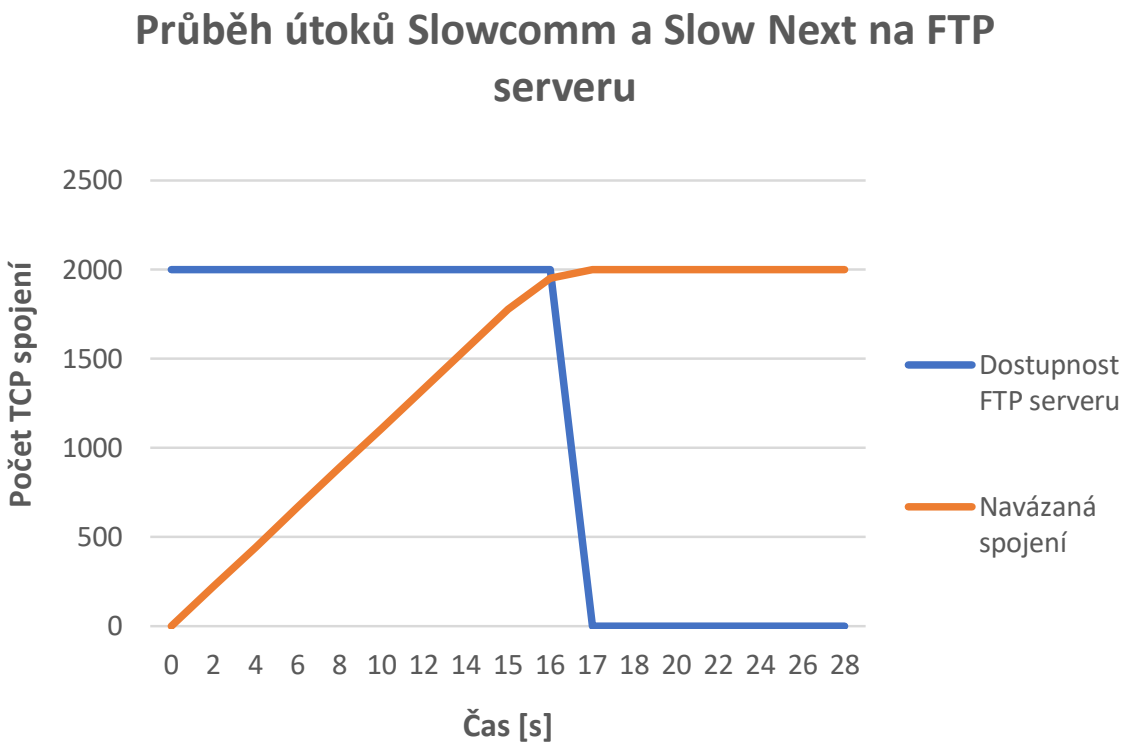
Příkaz pro spuštění útoku Slowcomm v nástroji SlowDoSGen na FTP server v příkazové řádce je následující:

```
python3 SlowDoSGen.py -a C -ip 10.10.0.2 -c 2000 -p 21
```

Při využití útoku **Slow Next** na FTP server jsou úvodní i udržovací pakety totožné: `USER <jmeno>`. I v tomto případě zamezení služby nastalo ihned po navázání dostatečného počtu spojení se serverem – ve výchozím stavu je v generátoru při útoku Slow Next iniciováno 10 vláken, každé vytváří 200 spojení, tzn. dohromady 2000 spojení.

Příkaz pro spuštění útoku Slow Next v nástroji SlowDoSGen na FTP server v příkazové řádce je následující:

```
python3 SlowDoSGen.py -a N -ip 10.10.0.2 -c 200 -tc 10 -p 21
```



Obr. 8.4: Průběh útoků Slowcomm a Slow Next na FTP serveru

V obou případech server zobrazí chybovou hlášku, nikoliv přihlašovací okno. Legitimní uživatel se tak nemůže připojit – bylo dosaženo útoku DoS.

8.4 Testování na protokolu SSH

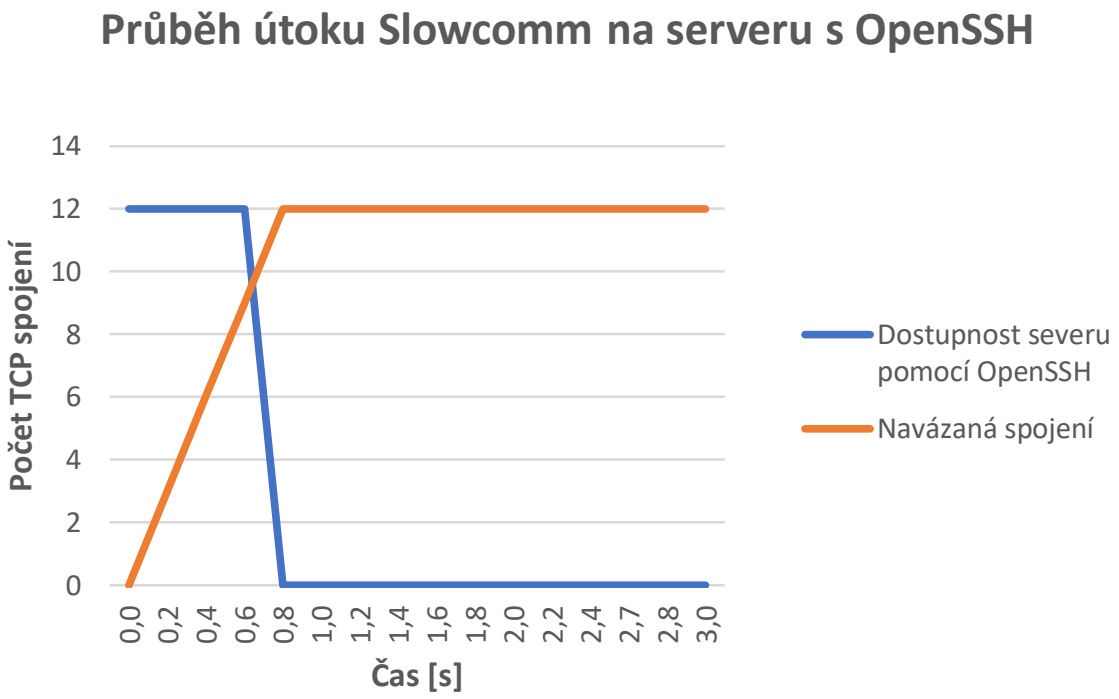
Testování bylo provedeno i na protokolu aplikační vrstvy SSH a nástroji OpenSSH. Provedeny byly obě varianty útoků v generátoru SlowDoSGen.

Útok **Slowcomm** byl úspěšný s následující konfigurací. Testováním bylo zjištěno, že pro znepřístupnění služby stačilo navázat pouhých 12 spojení, jak je vidět na obázku 8.5. Vysvětlení, proč stačil při útoku na protokol SSH násobně menší počet spojení než u ostatních protokolů může být ten, že již při sestavování zabezpečeného spojení jsou aplikovány postupy náročné na výpočetní kapacitu, jako například dohodnutí šifrovacího klíče pro relaci apod. Více v kapitole 1.3.

Příkaz pro spuštění útoku v nástroji SlowDoSGen v příkazové řádce:

```
python3 SlowDoSGen.py -a C -ip 10.10.0.2 -c 12 -p 22
```

Znovu se potvrdilo, že v případě útoku Slowcomm nezáleží na použitém payloadu. Stačí, že jsou příkazy nedokončené či nevalidní. Toto tvrzení se tak může aplikovat i na útoky tohoto typu u jiných protokolů aplikační vrstvy.



Obr. 8.5: Průběh útoku Slowcomm na serveru s OpenSSH

Útok **Slow Next** ovšem úspěšně proveden nebyl. Z definice tohoto typu útoku je nutná určitá kooperace serveru v podobě validních odpovědí na požadavky klienta. Byla otestována možnost implementace této taktiky hned při sestavování spojení a domluvy algoritmů a šifer, které budou použity zejména pro šifrování spojení. Při testování bylo zjištěno, že server jednoduše neumožňuje například opakování částí komunikace pro dosažení útoku DoS (jako je tomu v případě protokolu FTP, viz kapitola 1.2). Při obdržení například podporovaných verzí protokolu SSH vícekrát po sobě, server spojení ukončí. Tento přístup proto nebylo možné použít.

Jako druhý možný přístup se jeví v části samotné autentizace uživatele pomocí hesla, například opakované zasílání uživatelského jména. Bohužel ale protokol neumožňuje iniciovat poslání přihlašovacích údajů klientem (k zadání vyzve server), pro přihlášení uživatele je nastaven limit 3 pokusů a časový limit 60 sekund. Navíc se nepodařilo implementovat fázi vyměňování parametrů pro vytvoření klíče relace pro následné využití výše zmíněné metody. Nepodařilo se tedy nalézt cestu k úspěšnému provedení útoku Slow Next na protokolu SSH.

9 Testování účinnosti detekčního systému

Poslední kapitola této bakalářské práce se zabývá výsledky testování vytvořeného detekčního systému IDS ve všech variantách útoků testovaných v kapitole 8. Grafy vždy zobrazují průběh útoku a vyznačeny jsou i signatury, které byly detekčním systémem zaznamenány a vyhodnoceny jako útok. Oranžová křivka popisuje spojení navázaná útočníkem a modrá dostupnost serveru z hlediska potenciálně volných spojení pro legitimní uživatele při úspěšném útoku. Červená čárkovaná křivka značí hraniční hodnotu dané signatury a zelená tečkovaná křivka maximální počet spojení, který by byl server schopný zpracovat, kdyby po odhalení útoku byla provedena bezpečnostní opatření.

Jak bylo popsáno v kapitole 6.2, vytvořený detekční systém se spouští příkazem

`sudo python3 SlowDoS_IDS.py`.

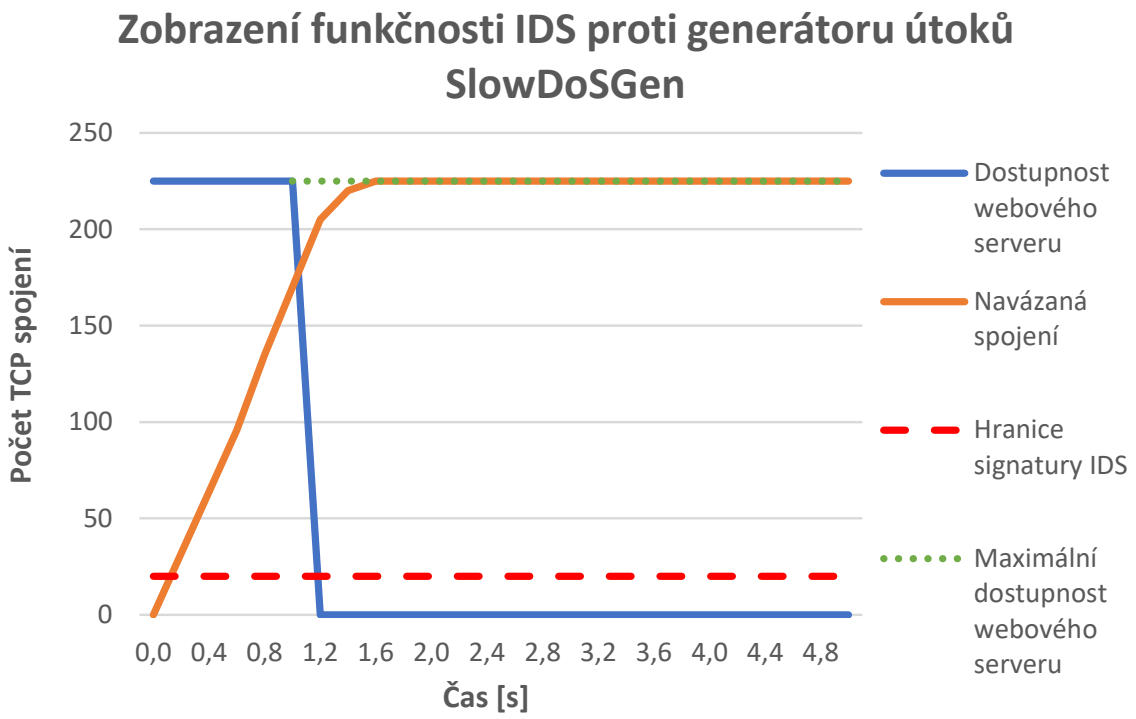
Dále bylo vybráno používané rozhraní, čímž se iniciovalo odposlouchávání sítě a detekční systém začal pracovat.

Výpis 9.1 ukazuje příklad obsahu log souboru po detekování útoku Slowcomm. Databáze podezřelých spojení je prázdná, protože útok probíhající pouze z IP adresy 10.10.0.4 byl detekován, IP adresa přidána do blacklistu, smazána z databáze a detekční systém ji dále ignoruje.

Výpis 9.1: Příklad obsahu log souboru

```
1  07-May-20 14:03:52 - CRITICAL: Attack Slowcomm on
       port 80 from IP 10.10.0.4 detected - too many
       connections from one IP
2  07-May-20 14:04:12 - INFO
3  Log of suspicious connections: {}
4  Log of blaclisted IP addresses: ['10.10.0.4']
```

Jelikož všechny útoky byly v testovacím prostředí testovány pouze z jednoho útočícího stroje, tzn. v nedistribuované formě, byla signatura detekující útok ve všech případech stejná – velké množství otevřených spojení z jedné IP adresy. Je tomu tak proto, že generátor útoků nestihne tak rychle vygenerovat dostatečný počet požadavků, které by následně detekční systém zachytil na základě jiných definovaných signatur. Pro ilustraci funkčnosti detekčního systému byl vybrán útok Slowcomm na webový server Apache, jakožto zástupce všech kombinací útoků testovaných generátorem SlowDoSGen na všech zkoumaných protokolech (viz kapitola 8). Při demonstraci útoku Slow Next by graf vypadal obdobně, hraniční hodnota signatury by byla nastavena stejně, pouze křivky samotného útoky by byly odlišné.



Obr. 9.1: Graf zobrazení funkčnosti IDS proti generátoru útoků SlowDoSGen

Graf na obrázku 9.1 reprezentuje data převzatá z průběhu útoku Slowcomm na webový server Apache (viz kapitola 8.1.2). V momentě, kdy se oranžová (navázaná spojení útočníkem) a červená čárkovaná křivka (hraniční hodnota signatury pro maximální počet spojení z jedné IP adresy, tzn. 20) protnou, je hraniční hodnota signatury překonána a útok je detekován. Je vyhlášeno varování do konzole a zapsán záznam do log souboru.

Celkově byla zjištěna bezchybná účinnost detektoru na všech testovaných protokolech aplikační vrstvy modelu ISO/OSI – HTTP, FTP a SSH, který je schopen odhalit útok zpravidla v rozmezí několika sekund. V momentě, kdy systém útok detekuje, zařadí zdrojovou IP adresu útočníka do blacklistu. Detektor tak může dále pracovat, přičemž útoky z této IP adresy již ignoruje, aby byl proces detekce přehlednější. Blacklist může být dále použit například síťovými administrátory k blokaci IP adres, ze kterých byly útoky vedeny, případně k jiným postupům.

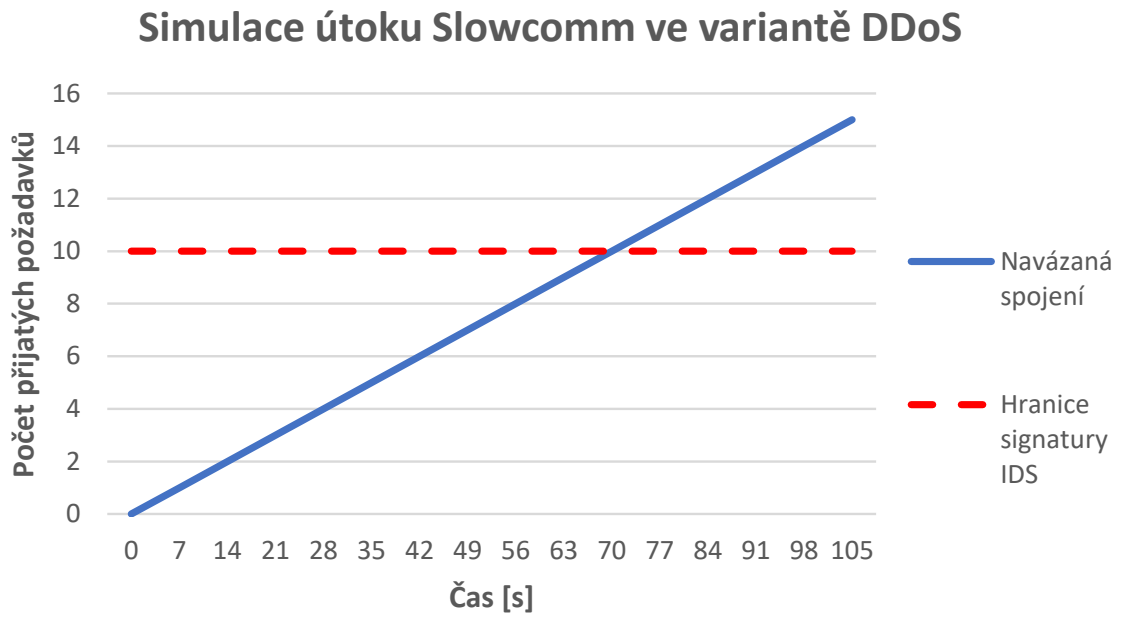
9.1 Simulace detekce distribuované formy útoku

Všechny testy útoků zmíněné v kapitole 9, byly v testovacím prostředí testovány pouze z jednoho útočícího stroje, tzn. v nedistribuované formě. Pro důkaz funkčnosti detekčního systému i při útoku distribuovanou formou byl simulován datový

provoz útoku opět z jednoho stroje, ale v redukované míře, jakožto například útočníkem používaných botů (viz kapitola 2), kteří se v kooperaci se stovkami dalších snaží dosáhnout efektu útoku DoS. Útočník v této simulaci navazuje pouze jedno spojení z každého botu pomocí generátoru SlowDoSGen (viz kapitola 4). Detekce takto vedeného útoku je časově náročnější, protože je nejdříve nutné zpracovat vetší množství dat. Záleží ale na konkrétní konfiguraci útoku.

9.1.1 Detekce útoku Slowcomm

V první části simulace útoku DDoS je znázorněn síťový provoz jednoho bota provozovaného útočníkem, za použití útoku Slowcomm na službu OpenSSH.



Obr. 9.2: Simulace útoku Slowcomm ve variantě DDoS

Jak je vidět na obrázku 9.2, zde detekční systém v síťovém provozu zaznamenal, že bylo posláno více než 10 v tomto případě neukončených požadavků (požadavky neobsahovaly ukončovací znak `\r\n`) – průnik modré a čárkované červené křivky.

Výpis 9.2: Záznam log souboru – detekce útoku Slowcomm

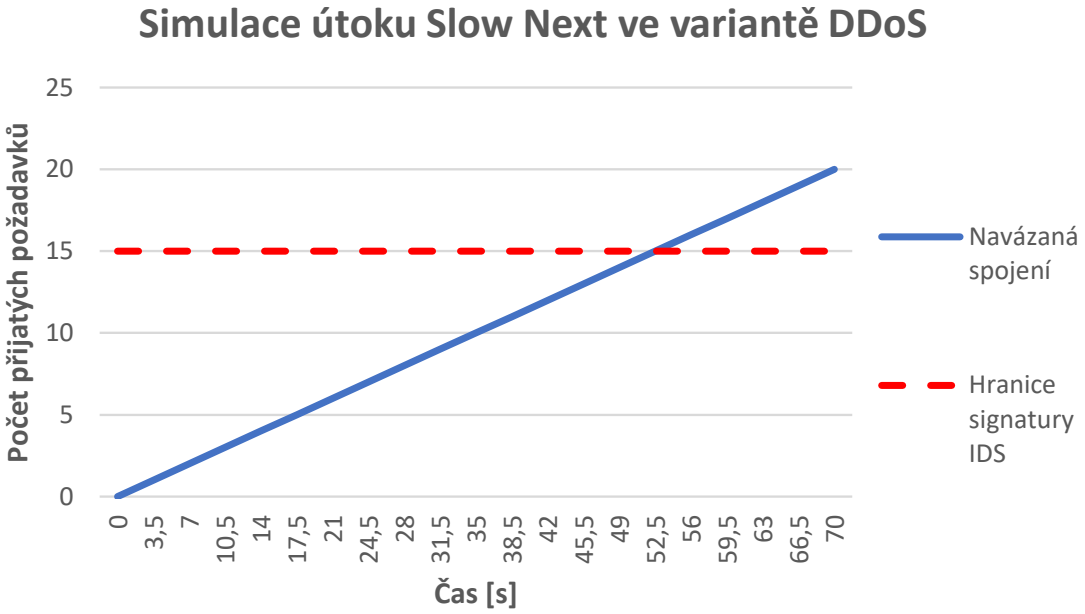
```
1   07-May-20 14:53:07 - CRITICAL: Attack Slowcomm on
      port 22 from IP 10.10.0.4 detected - too much time
       for finishing request in one connection
```

Systém dále ověřil čas, po který jsou v rámci zkoumaného spojení posílány tyto neukončené požadavky. Jelikož se jednalo o více než 15 sekund (konkrétně 70), což

je hraniční hodnota signatury pro maximální počet požadavků v rámci jednoho spojení a doby jeho trvání, byl detekován útok Slowcomm. Log detekovaného útoku je zobrazen ve výpisu 9.2.

9.1.2 Detekce útoku Slow Next

Ve druhé části simulace útoku DDoS je znázorněn síťový provoz jednoho bota provozovaného útočníkem na FTP server vsftpd, ale za použití útoku Slow Next.



Obr. 9.3: Simulace útoku Slow Next ve variantě DDoS

Na obrázku 9.3 je znázorněno zachycení signatury stejného či podobného časového intervalu mezi přijímanými ukončenými HTTP požadavky. Detekční systém v tomto případě zachytil 15. přijímaný ukončený požadavek a ověřil jejich časové rozestupy, kdy požadavky přicházely minimálně se 2 sekundovým odstupem a hodnota tohoto časového odstupu se lišila o maximálně 0,1 sekundy. Je tak zaznamenána určitá strojová přesnost zasílání požadavků, splnění parametrů definované signatury a detekován útok Slow Next. Výpis log souboru je zobrazen ve výpisu 9.3.

Výpis 9.3: Záznam log souboru – detekce útoku Slow Next

```
1   07-May-20 14:27:42 - CRITICAL: Attack Slow Next on
      port 21 from IP: 10.10.0.4 detected - signature
      response time found
```

9.1.3 Hodnocení detekce distribuované formy útoků

Jak si bylo možné všimnout v kapitolách 9 a 9.1, zatímco detekce útoku z jednoho počítače proběhne do jedné vteřiny, detekce distribuované formy útoku může být i záležitostí jedné minuty. Záleží na tom, jak dlouhý je interval mezi zasíláním požadavků. Detekční systém musí nejdříve shromáždit více dat, aby bylo možné s vysokou pravděpodobností podle chování jednotlivých spojení říci, že se jedná o útok. Jelikož jsou požadavky posílány s poměrně velkým odstupem mezi sebou, může tato operace zabrat určitý čas. V této situaci proto útoky v distribuované formě vždy na několik vteřin dosáhnou efektu DoS.

V případě útoku Slowcomm trvala detekce zhruba 63 sekund, při útoku Slow Next 52 sekund. To je zároveň doba, kdy byl server nedostupný pro legitimní uživatele. V nejlepším scénáři by toto vůbec nemělo nastat. Bohužel ale nebyla nalezena žádná metoda vhodná pro splnění tohoto cíle. Nabízela se možnost snížení hraničních hodnot některých signatur, např. minimální hodnoty přijatých požadavků. Tento postup ale samozřejmě výrazně ovlivňuje spolehlivost detekce opravdových útočníků. Jednoduše by se některá z podezřelých IP adres mohla zaměnit s legitimním uživatelem s pouze pomalým připojením k síti. V případě útoku Slow Next navíc tento postup vůbec možný není, protože síťový provoz generovaný útočníkem a legitimním uživatelem jsou prakticky totožné.

Alternativou pro možnost rychlejší detekce by bylo vytvoření detekčního systému s mnohem hlubší analýzou síťového provozu, který by například zkoumal podobnost požadavků napříč všemi podezřelými spojeními z různých IP adres. Legitimní uživatel by mohl mít požadavky jiné, zatímco požadavky z útočníkova botnetu (viz kapitola 2) by teoreticky mohly být stejné. Zde ovšem nastává další problém. Co kdyby útočník posílal požadavky zcela náhodně, například z nějakého předem vygenerovaného seznamu. To by situaci znovu několikanásobně ztížilo. Opět bychom dospěli pouze k možnému zvýšení pravděpodobnosti probíhajícího útoku, ale ne k přesnému určení původce.

Další možnost, která připadá v úvahu, je manuální snížení velikosti timeoutu pro perzistentní spojení, která je ve výchozím stavu nastavena na dlouhých 5 sekund. To by mohlo ztížit útočníkovi situaci a zároveň mnohonásobně zvýšit síťový provoz, což by mohlo pomoci k odhalení útoku. V případě webového serveru Apache by bylo možné více přizpůsobit bezpečnostní moduly, ovšem to by mohlo vést k omezení legitimních uživatelů s pomalým připojením k síti. Zkoumané pomalé DoS útoky se tak jeví jako velice účinné, protože jinak je nejspíše není možné odhalit. Jejich detekce v distribuované formě je pro vytvořený detekční systém velice obtížná. Bylo by nutné použít detekční systémy založené na neuronových sítích, fuzzy systémech nebo umělé inteligenci [4].

10 Závěr

Cílem teoretické části této bakalářské práce bylo prostudování problematiky pomalých DoS útoků nezávislých na aplikační vrstvě, konkrétně útoků Slowcomm, SlowReq a SlowNext, a zároveň vytvořit i modely těchto útoků. Cílem praktické části bylo vytvořit testovací prostředí s nástrojem pro generování výše zmíněných útoků a otestovat jak jeho funkčnost, tak i jeho účinnost na vybraných serverech. Jelikož generátor těchto útoků nebyl z žádných zdrojů dostupný, byl v rámci praktické části naprogramován. Druhým cílem bylo vytvořit systém schopný tyto útoky v síti detekovat.

V teoretické části práce byly představeny DoS útoky a jejich rozdělení se zvláštním zaměřením na poměrně novou skupinu z nich, tzv. pomalé DoS útoky Slowcomm, Slow Next a SlowReq. Při analýze útoků bylo zjištěno, že dva z nich jsou v podstatě totožné, a dále se pracovalo pouze s útoky Slowcomm a Slow Next. Následně byly představeny modely útoků, podle kterých se prováděla jejich implementace. Byly zmíněny konkrétní použité parametry, které mají v algoritmech útoků rozhodující vliv na jeho funkci a úspěšnost, včetně bezpečnostních opatření webového serveru Apache. Dále byl navržen systém detekce průniku proti zkoumaným pomalým DoS útokům fungující na základě detekce definovaných signatur, podle kterých program úspěšně rozeznává probíhající útoky od legitimního provozu sítě.

V rámci praktické části práce bylo vytvořeno testovací prostředí, sestávající ze tří virtuálních strojů pro útočníka, legitimního uživatele a serveru. Kvůli bezpečnosti byly všechny stroje umístěny do interní sítě bez přístupu k internetu a k hostujícímu počítači. Na základě modelů navržených v teoretické části vznikl nástroj SlowDoSGen vytvořený v jazyce Python, jakožto generátor útoků Slowcomm a Slow Next. Práce obsahuje i návod pro pochopení jeho obsluhy a prezentuje parametry, kterých je možné využít při inicializaci útoku. Tento nástroj byl následně využíván pro testování.

Všechny útoky byly testovány na protokolech HTTP, FTP a SSH aplikační vrstvy referenčního modelu ISO/OSI, které byly popsány v úvodu práce. U protokolu HTTP bylo testování rozděleno do dvou fází. První fáze se snažila docílit útoku DoS na webový server Apache bez jakékoliv ochrany, tzn. bez integrovaného bezpečnostního modulu ReqTimeout. Ve druhé fázi byl modul ReqTimeout zapnut. Ukázalo se, že ani jednomu z útoků nedělá tento systém výraznější problémy. Tento modul, cílený hlavně proti útokům typu Slowcomm, se neosvědčil jako plně funkční, je spíše omezující z hlediska narušení dlouhodobé účinnosti útoku. V dalším kroku bylo provedeno testování i na webových serverech Nginx, lighttpd a IIS od společnosti Microsoft. Byly zde komentovány důvody, proč jsou servery IIS a částěčně i Nginx vůči zkoumaným útokům imunní a jaké jsou jejich bezpečnostní výhody.

Následně byly útoky otestovány i na protokolu FTP a SSH, aby byla dokázána jejich nezávislost na aplikačních protokolech. Útoky sice nefungovaly se stejnými payloady (úvodními a udržovacími požadavky), po jejich úpravě byly ale útoky na servery úspěšné, až na útok Slow Next na službu OpenSSH, protože se nezdařilo nalézt a vhodně implementovat část protokolu ke splnění definice útoku Slow Next. Z tohoto faktu můžeme usoudit, že nezávislost pomalých DoS útoků spočívá v jejich samotné implementaci, tj. principu, jakým útočí – payloady musí být upraveny podle služby, na kterou útočník cílí.

Dále byl otestován i vytvořený systém IDS. Fungoval spolehlivě a bezchybně a odhalil útoky na všech testovaných službách. Ze záznamů detekčního systému lze získat potřebná data jako IP adresu, port a důvod jejich zachycení. Na základě těchto informací se mohou dále provést potřebné kroky k zastavení útoku v systému prevence průniku, například blokace IP adresy.

Závěrem lze konstatovat, že díky funkčnosti všech útoků na webový server Apache, FTP server vsftpd a částečně i na službu OpenSSH splňuje tato práce svůj účel. Pomocí implementovaných útoků můžeme téměř okamžitě dosáhnout útoku DoS a odepřít službu legitimním uživatelům na takřka neomezeně dlouhou dobu. Po vypnutí generátoru začal cílený server opět poskytovat své služby. Vytvořený detekční systém na druhou stranu umožňuje tyto útoky bezpečně detekovat. Celkově tak tato práce poskytuje skvělou základnu pro získání vědomostí ohledně útoků DoS, konkrétněji jejich pomalé varianty, a zároveň si je i vyzkoušet v praxi.

Cílem dalšího vývoje generátoru SlowDoSGen by mohlo být rozšíření podpory dalších pomalých DoS útoků, případně doplnění vhodných payloadů pro další protokoly aplikační vrstvy. U detekčního systému by mohlo jít o optimalizaci pro další protokoly, případný návrh jiného druhu detekčního systému, například pomocí neuronových sítí.

Literatura

[1] CAMBIASO, Enrico, Gianluca PAPALEO a Maurizio AIELLO. Taxonomy of Slow DoS Attacks to Web Applications. *Recent Trends in Computer Networks and Distributed Systems Security* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, s. 195–204 [cit. 2019–11–29]. Communications in Computer and Information Science. DOI: 10.1007/978 − 3 − 642 − 34135 − 920. ISBN 978-3-642-34134-2. Dostupné z URL: <http://link.springer.com/10.1007/978-3-642-34135-9_20>.

[2] CAMBIASO, Enrico, Gianluca PAPALEO, Giovanni CHIOLA a Maurizio AIELLO. Slow DoS attacks: definition and categorisation. *International Journal of Trust Management in Computing and Communications*. 2013, roč. 1 (3/4), 20 stran. DOI: 10.1504/IJTMCC.2013.056440. ISSN 2048-8378. Dostupné z URL: <http://www.inderscience.com/link.php?id=56440>.

[3] CAMBIASO, Enrico, Gianluca PAPALEO a Maurizio AIELLO. Slowcomm: Design, development and performance evaluation of a new slow DoS attack. *Journal of Information Security and Applications*. 2017, roč. 35, s. 23–31. DOI: 10.1016/j.jisa.2017.05.005. ISSN 22142126. Dostupné z URL: <https://linkinghub.elsevier.com/retrieve/pii/S2214212616300680>.

[4] CAMBIASO, Enrico, Gianluca PAPALEO, Giovanni CHIOLA a Maurizio AIELLO. Designing and Modeling the Slow Next DoS Attack. *International Joint Conference. Cham: Springer International Publishing*. 2015, strany 249–259. Advances in Intelligent Systems and Computing. DOI: 10.1007/978-3-319-19713-5_22. ISBN 978-3-319-19712-8. Dostupné z URL: <http://link.springer.com/10.1007/978-3-319-19713-5_22>.

[5] AIELLO, Maurizio, Gianluca PAPALEO a Enrico CAMBIASO. SlowReq: A Weapon for Cyberwarfare Operations. Characteristics, Limits, Performance, Remediations. *International Joint Conference SOCO'13-CISIS'13-ICEUTE'13. Cham: Springer International Publishing*. 2014, s. 537–546. Advances in Intelligent Systems and Computing. DOI: 10.1007/978-3-319-01854-6_55. ISBN 978-3-319-01853-9. Dostupné z URL: <http://link.springer.com/10.1007/978-3-319-01854-6_55>.

[6] FIELDING, R. *RFC 7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. ISSN: 2070-1721. Uděleno 2014.Dostupné z URL: <https://tools.ietf.org/html/rfc7231>.

[7] BERNERS-LEE, T. *RFC 1945: Hypertext Transfer Protocol – HTTP/1.0*. Uděleno 1996. Dostupné z URL: <https://tools.ietf.org/html/rfc1945>.

[8] MIRKOVIC, Jelena, Sven DIETRICH, David DIETRICH a Peter REIHER. *Internet denial of service: attack and defense mechanisms*. Upper Saddle River, N.J.: Prentice Hall Professional Technical Reference, 2004. ISBN 01-314-7573-8.

[9] POSTEL, J. a J. REYNOLDS. *RFC 959: FILE TRANSFER PROTOCOL (FTP)*. Uděleno 1985. Dostupné z URL: <https://tools.ietf.org/html/rfc959>.

[10] YLONEN, Tatu. SSH (Secure Shell). *Ssh.com* [online]. Helsinki: Tatu Ylönen, 1995 [cit. 2020-05-16]. Dostupné z URL: <https://www.ssh.com/ssh/>.

[11] The most important steps to take to make an Apache server more secure. *DreamHost* [online]. Los Angeles: New Dream Network, 2019 [cit. 2019–12–02]. Dostupné z URL: <https://bit.ly/2E5MIfX>.

[12] Documentation. *ModSecurity* [online]. Chicago: Trustwave Holdings, 2018 [cit. 2019–12–02]. Dostupné z URL: <https://modsecurity.org/documentation.html>.

[13] MARTINÁSEK, Zdeněk. *Systémy IDS a IPS: Bezpečnost ICT2 [online prezentace]*. Vysoké učení technické v Brně, 2018 [cit. 2020-05-05].

[14] DUBROY, Patrick. How many tabs do people use? *Patrick Dubroy* [online]. 2009 [cit. 2020-05-09]. Dostupné z URL: <https://dubroy.com/blog/how-many-tabs-do-people-use-now-with-real-data/>.

[15] Netplan: The network configuration abstraction renderer. *Canonical* [online]. Londýn: Canonical, 2018 [cit. 2019–12–06]. Dostupné z URL: <https://netplan.io/>.

[16] December 2018 Web Server Survey. *Netcraft* [online]. Bath: Netcraft, 2018 [cit. 2019-11-29]. Dostupné z URL: <https://news.netcraft.com/archives/2018/12/17/december-2018-web-server-survey.html>.

[17] Apache HTTP Server Documentation. *Apache HTTP server project* [online]. Forest Hill, Maryland: Apache Software Foundation, 1999 [cit. 2019-11-25]. Dostupné z URL: <http://httpd.apache.org/docs/>.

[18] KROCZEK, Grzegorz. NGINX vs Apache – power engine asynchronous and even-driven server. *Independent Software Developer: Grzegorz Kroczek ϕ – maxprog@maxprog.net.pl* [online]. Grzegorz Kroczek, 2015 [cit. 2020-05-09]. Dostupné z URL: <https://bit.ly/2Lkj7ml>.

[19] GARRETT, Owen. Inside NGINX: How We Designed for Performance & Scale. *NGINX* [online]. Seattle: F5 Networks, 2015 [cit. 2020-05-09]. Dostupné z URL: <https://www.nginx.com/blog/inside-nginx-how-we-designed-for-performance-scale/>.

[20] TANWAR, Shakti. *Hands-On Parallel Programming with C# 8 and .NET Core 3* [online]. Birmingham: Packt Publishing, 2019, s. 251 [cit. 2020-05-09]. ISBN 978-1-78913-241-0. Dostupné z URL: <https://bit.ly/3duOZkj>.

[21] KNESCHKE, Jan. Lighttpd. *Lighttpd - fly light* [online]. [2015] [cit. 2020-05-09]. Dostupné z URL: <https://www.lighttpd.net/>.

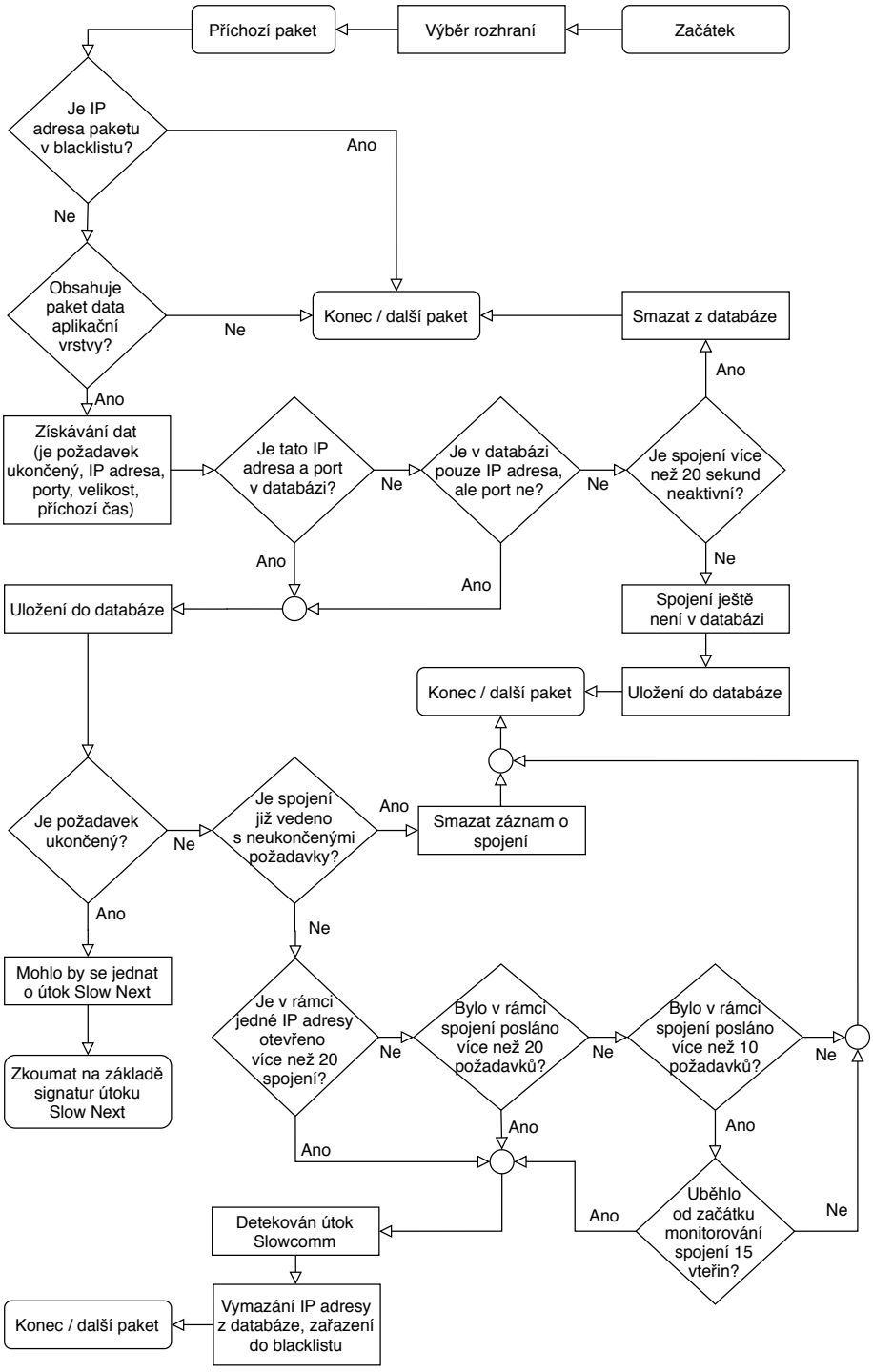
Seznam symbolů, veličin a zkratek

HTTP	Hypertext Transefer Protocol
TCP	Transmission Control Protocol
URL	Uniform Resource Locator
FTP	File Transfer Protocol
SSH	Secure Shell
DoS	Denial of Services
DDoS	Distributed Denial of Services
IDS	Intrusion detection system
IPS	Intrusion prevention system

Seznam příloh

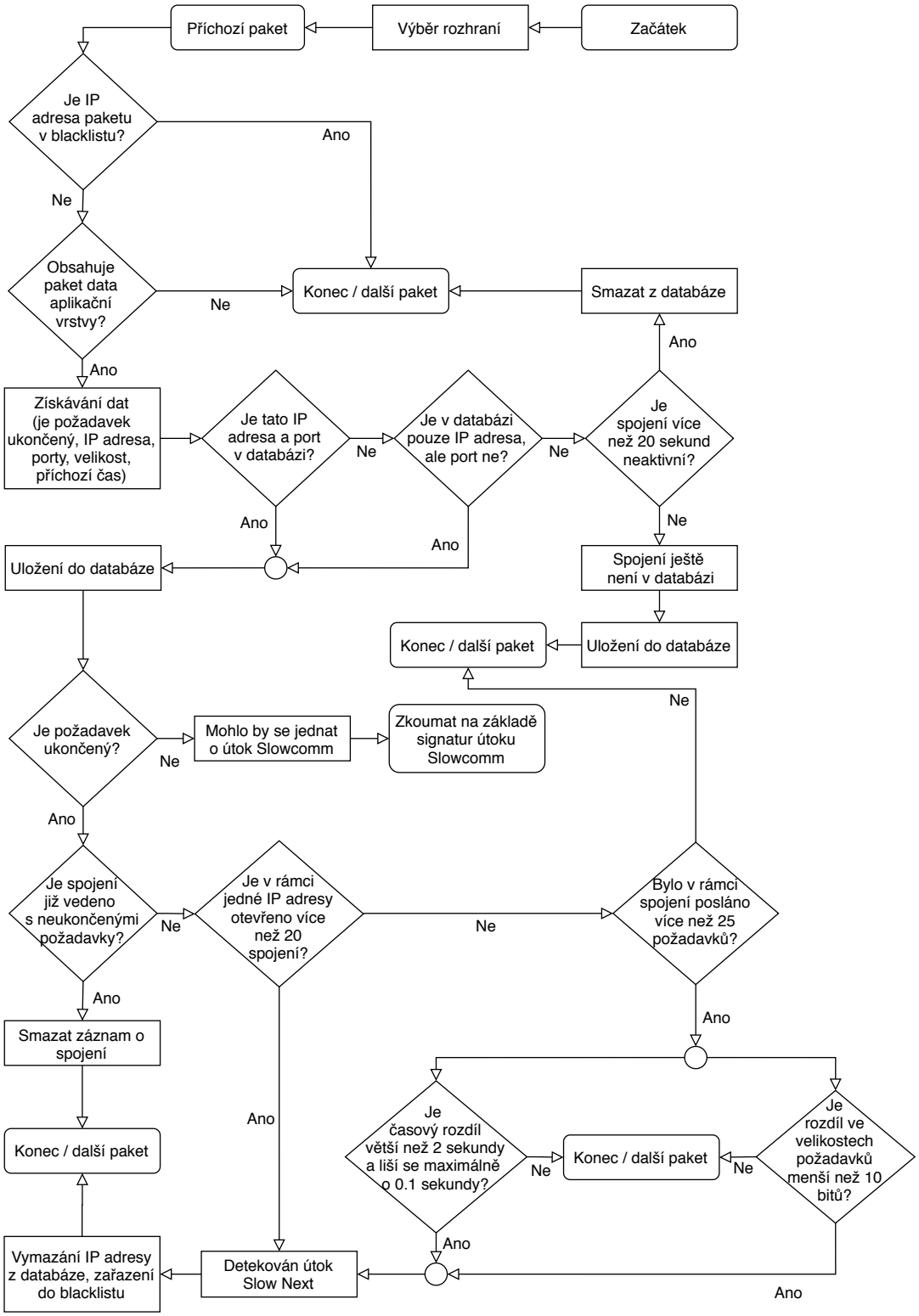
A Vývojový diagram detektoru

A.1 Algoritmus detekce útoku Slowcomm



Obr. A.1: Vývojový diagram detektoru útoků typu Slowcomm

A.2 Algoritmus detekce útoku Slow Next



Obr. A.2: Vývojový diagram detektoru útoků typu Slow Next

B Manuál ke generátoru SlowDoSGen

Python skript SlowDoSGen.py je generátor pomalých DoS útoků Slowcomm a Slow Next, sloužících k zamezení služby na protokolech aplikační vrstvy. Jsou k tomu využity tzv. sokety, které propojí počítač útočníka s cíleným serverem. Tyto útoky mají 3 fáze:

1. Sestavení maximálního počtu spojení se serverem pomocí tzv. úvodních požadavků.
2. Udržení spojení v aktivním stavu pomocí tzv. udržovacích požadavků.
3. Detekce uzavřených spojení a jejich nahrazení novými.

Více informací naleznete v bakalářské práci:
RICHTER, Dominik. *Slow rate DoS útoky nezávislé na protokolu aplikační vrstvy.* Brno, 2020, 69 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Marek Sikora.

B.1 Jak skript nainstalovat a spustit

Stáhněte si skript SlowDoSGen.py. Spouští se pomocí příkazové řádky příkazem `python` a jména generátoru s přidruženými parametry. Pro spuštění je nutná verze Python 3.0 a vyšší. Pro zobrazení nápovědy proveďte příkaz:

```
python3 SlowDoSGen.py -h.
```

B.2 Konfigurace útoku

Útoky je možné konfigurovat pomocí následujících parametrů.
Povinné parametry:

- `-a <str>`
  Tento parametr slouží k rozlišení mezi jednotlivými útoky. `-a C` pro útok Slowcomm a `-a N` pro útok Slow Next.
- `-ip <str>`
  Slouží k zadání cílové IP adresy.
- `-p <int>`
  Používá se k zadání cílového portu.
- `-c <int>`
  Určuje požadovaný počet spojení s cílem.

Volitelné parametry:

- `-l <payload.txt>`
  Slouží pro specifikaci vlastního payloadu pomocí textového souboru, pojmenovaného payload.txt, ve stejném adresářovém umístění jako samotný skript generátoru. V případě útoku Slowcomm útočník zapíše požadavek na první řádek souboru. V případě útoku Slow Next útočník na první řádek zapíše úvodní požadavek a na druhý řádek zapíše udržovací požadavek. Vždy včetně znaků odřádkování, např.:

  `HEAD / HTTP/1.1\r\nHost: 192.168.0.155\r\n\r\n`.

- `-tc <int>`
  Používá se pouze v případě útoku Slow Next. Určuje množství vláken, které útočník potřebuje vytvořit. Výchozí hodnota je 5 vláken.
- `-t <float>`
  Určuje timeout pro posílání požadavků – jak často jsou odesílány v cyklu. Výchozí hodnotou útoku Slowcomm je 7 sekund. Pro útok Slow Next 3,5 sekundy.
- `-t2 <float>`
  Pouze pro útok Slow Next. Specifikuje, v jakém časovém intervalu jsou vlákna spouštěna. Výchozí hodnotou je 1 sekunda.

B.3 Příklady použití na webovém serveru

Slowcomm:

```
python3 SlowDoSGen.py -a C -ip 10.10.0.2 -c 225 -p 80
```

Slow Next:

```
python3 SlowDoSGen.py -a N -ip 10.10.0.2 -c 140 -p 80
```

C Obsah příloh

Adresář s přílohami bakalářské práce obsahuje 2 skripty psané v jazyce Python v podadresáři `Skripty`. Pro jejich spuštění je nutná verze Python 3.0 a vyšší. Skript `SlowDoSGen.py` obsahuje zdrojový kód ke generátoru pomalých útoků DoS. Druhý skript `SlowDoS_IDS.py` obsahuje zdrojový kód k systému detekce průniku útoků prováděných pomocí generátoru SlowDoSGen. Adresář dále obsahuje elektronickou verzi bakalářské práce a manuál ke generátoru SlowDoSGen.

- `/` ..... kořenový adresář příloh
  - `Skripty` ..... spustitelné skripty v jazyce Python
    - `SlowDoSGen.py`
    - `SlowDoS_IDS.py`
  - `BakalarskaPrace.pdf` ..... elektronická kopie této práce
  - `SlowDoSGen_manual.pdf` ..... manuálové stránky generátoru SlowDoSGen