VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

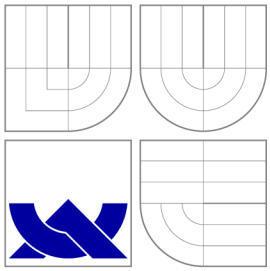
DETEKCE POMALÝCH DOS ÚTOKŮ NA HTTP

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

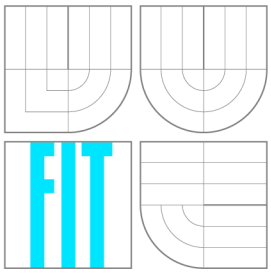
AUTOR PRÁCE
AUTHOR

PATRIK JAKUBÍČEK

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DETEKCE POMALÝCH DOS ÚTOKŮ NA HTTP

DETECTION OF SLOW HTTP DOS ATTACKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PATRIK JAKUBÍČEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VÁCLAV BARTOŠ

BRNO 2014

Abstrakt

Tato bakalářská práce se zabývá detekcí Slowloris útoku. Na základě poznatků je vytvořen detekční modul pro systém Nemea, který provádí detekci útoku pomocí záznamů o tocích. Testováním bylo ověřeno, že modul dokáže pracovat v reálném provozu a poměrně úspěšně detekovat Slowloris útok.

Abstract

This thesis deals with the detection of Slowloris attack. Based on the findings a detection module for Nemea system is implemented. It analyzes flow records and performs attack detection. Tests have verified that the module can work in real deployment and detect Slowloris attack quite successfully.

Klíčová slova

Slow HTTP DoS, slowloris, DoS, Nemea, detekce útoku, síťové útoky

Keywords

Slow HTTP DoS, slowloris, DoS, Nemea, attack detection, networks attacks

Citace

Patrik Jakubíček: Detekce pomalých DoS útoků na HTTP, bakalářská práce, Brno, FIT VUT v Brně, 2014

Detekce pomalých DoS útoků na HTTP

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Václava Bartoše.

.....
Patrik Jakubíček
15. května 2014

Poděkování

Děkuji Ing. Václavu Bartošovi za vedení práce, vstřícný přístup a cenné rady v celém průběhu vytváření práce. Samozřejmě také děkuji rodině, přítelkyni a přátelům za podporu.

Obsah

Seznam obrázků

Kapitola 1

Úvod

V dnešní době, kdy počítače jsou neodmyslitelnou součástí většiny domácností, řídí nespočet věcí v našem okolí a denně na jejich práci spoléháme, je velmi důležité abychom dokázali kontrolovat hrozby, které současně s tím přichází. Aby tyto stroje mohly pracovat, téměř vždy potřebují komunikovat s jinými stroji a pokud je využíváme my, například jako osobní počítač, také jistě požadujeme být online. K tomu, aby počítače, ať už slouží k jakémukoliv účelu, mohly komunikovat, slouží další počítače (servery), které nám zprostředkovávají komunikaci, zasílají důležitá data, která jsou na nich uložena a tak dále. Celá tato síť je pak vystavena mnoha hrozbám, které mohou mít za následek bezvýznamný problém anebo velice vážné dopady na lidský život. Abychom předešli těmto následkům, je nutné zavčas detekovat a eliminovat tyto hrozby, což je velice obtížné, protože různorodost útoků na servery samotné, či data proudící mezi počítači, je velká a neustále se vyvíjející.

Jelikož popularita útoků na počítačové sítě prudce stoupá, možná i z toho důvodu, že provést takový útok už není záležitostí zkušených počítačových pirátů, ale stačí průměrné znalosti práce s počítačem a přístup k internetu, je nutné věnovat se tomuto problému více. Existuje mnoho typů útoků na počítačové sítě a hlavní odlišností jsou zřejmě cíle útoku. Například zda útočník chce získat nějaká důležitá data, podvrhnout data anebo vyřadit nějakou službu z provozu, což je asi nejznámější variantou, více o těchto útocích lze nalézt v kapitole 2. Tyto útoky typu Denial of Service (DoS) v posledních letech dostali do povědomí uživatelů skupiny tzv. hacktivistů, nejznámější z nich je asi skupina Anonymous, jež se podíleli na mnoha mediálně velmi sledovaných útocích v posledních letech. Pomocí útoků se snažili upozornit na problémy, které v té době byly řešeny a vyjádřit k nim svůj názor. Ať je důvod či cíl útoku jakýkoliv, použití vhodné formy DoS útoku dokáže zvýšit šanci na úspěch.

Většina uživatelů počítačů si pod pojmem DoS útok představí velké množství počítačových pirátů útočících na jeden server. Pokud jde o jméno DoS, pak mají přibližně správné představy, ale pokud se na DoS útoky podíváme, jako na typ útoku s cílem vyřadit nějakou službu z provozu, zjistíme, že zde existuje velké množství variant DoS útoků. Všechny tyto varianty mají snahu o vyřazení z provozu nějaké služby a liší se v tom, na jakou část oběti útočí a také způsobem provedení. Jednou z těchto variant je Slow DoS útok na http, známý také jako Slowloris, jenž je hlavním tématem mé práce. Celá problematika toto útoku je detailněji vysvětlena dále v kapitole 3.

Hlavním cílem mé práce je vytvořit detektor Slowloris útoku ve formě modulu pro systém Nemea. Na tento modul jsou kladeny specifické požadavky, jelikož by měl pracovat bez ustání. Dále je nutné, aby v reálném čase zvládal zpracovávat velké objemy dat bez větších časových prodlev, ale také bezproblémové zpracování předem uložených dat, například pro

pozdější vyhodnocení provozu na síti. Jelikož objem dat proudící skrz modul je velký, bude nutné vyřešit vyfiltrování dat, práci s nimi tak, aby nedocházelo k hromadění dat a tím paměťovým i časovým ztrátám. Případným výstupem modulu při detekování útoku bude zpráva o probíhajícím útoku a po jeho ukončení i o celém průběhu útoku.

Moje práce je spojena se systémem Nemea a protože se jedná o poměrně rozsáhlý a zajímavý systém, a také to jistě pomůže k pochopení některých rozhodnutí při návrhu a implementaci, rád bych čtenáře s tímto projektem seznámil a to v kapitole 4. Podrobnějším návrhem modulu se zabývá kapitola 5, kde se pokusím objasnit, jak jsem se připravoval na implementování a jakou formu práce jsem zvolil. Způsob implementace, problémy, které bylo v té fázi vývoje potřebné řešit, jsou vysvětleny v kapitole 6. Předposlední kapitola se zabývá testováním modulu, jež bylo vzhledem k požadavkům a typu práce poměrně časově náročné. V této kapitole 7 je také možné shlédnout výsledky a vyhodnocení některých testů. V poslední kapitole 8 je uvedeno shrnutí a vyhodnocení celé mé práce.

Kapitola 2

Síťové útoky a typ Denial of Service

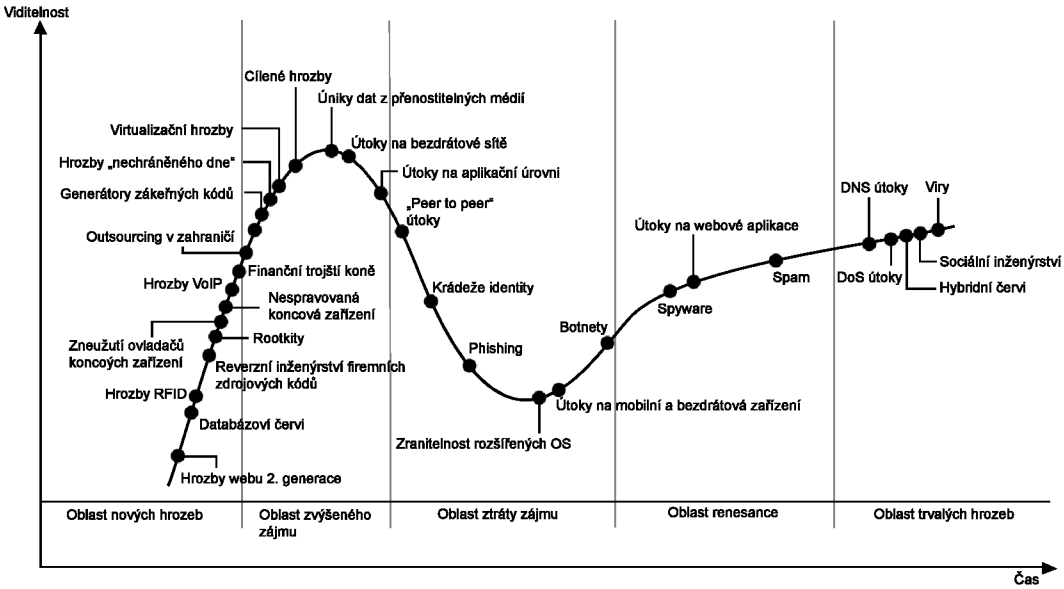
Protože problematika útoků na počítačové sítě je a zřejmě vždy bude důležitým tématem v oblasti výpočetní techniky, je tato kapitola věnována síťovým útokům. Jsou zde uvedeny důvody k použití útoků a jejich základní rozdělení podle několika kritérií.

A jelikož základem mojí práce je útok typu Denial of Service, tak bych rád dále objasnil základní problematiku těchto útoků. Tyto útoky, i přesto, že jsou známé již velmi dlouhou dobu, se staly neodmyslitelnou součástí počítačové kriminality. Důvodů může být hned několik, jejich velká efektivita, v dnešní době i snadný způsob provedení, také to, že cena ochrany před těmito útok může být velká, dále fakt, že vnikají nové modifikace, proti kterým často zpočátku nevíme jak se bránit.

2.1 Síťové útoky

K celé oblasti informačních technologií vždy patřily, patří a nadále patřit budou počítačové útoky[11]. Toto spojení je naprosto jisté, protože zde vždy budou lidé, kteří se budou snažit prolomit či využít bezpečnostní chyby systémů, ať už za účelem krádeže, pobavení, poškození oběti, zvýšení postavení nebo z jiného důvodu. Proto je nutné věnovat nejméně stejné úsilí detekci a eliminování těchto hrozeb, jako vývoji nových technologií. Dále je uvedeno shrnutí informací o počítačových útocích, celý článek zabývající se tématikou počítačové kriminality lze nelézt zde[8].

V poslední době se velmi zvýšila popularita využití některých útoků k vyjádření nespokojenosti či nesouhlasu, k čemuž pomohly, do té doby méně známé skupiny hacktivistů a velký zájem médií o tyto problémy. Potom takový útok na známý a velmi využívaný server, například zpravodajského typu, který se na čas stal z důvodu útoku nedostupný nebo se na jeho webových stránkách objevily hesla a prohlášení útočníků, vyvolá zájem médií, tím i popularitu. Dále už stačí, aby se útočník či skupina útočníků, ať už naprosto anonymně či pod názvem, které pomůže jejich popularizaci, vyjádřila k provedení tohoto útoku a s tím sdělila i důvod útoku. S tímto důvodem často sympatizuje velký dav lidí a další zájemci o tuto formu projevu jsou na světě. Například těmto formám útoků slouží právě útoky typu DoS, i jeho méně známá varianta Slow HTTP DoS.



Obrázek 2.1: Zobrazení útoků pomocí tzv. Hype Cycle.[1]

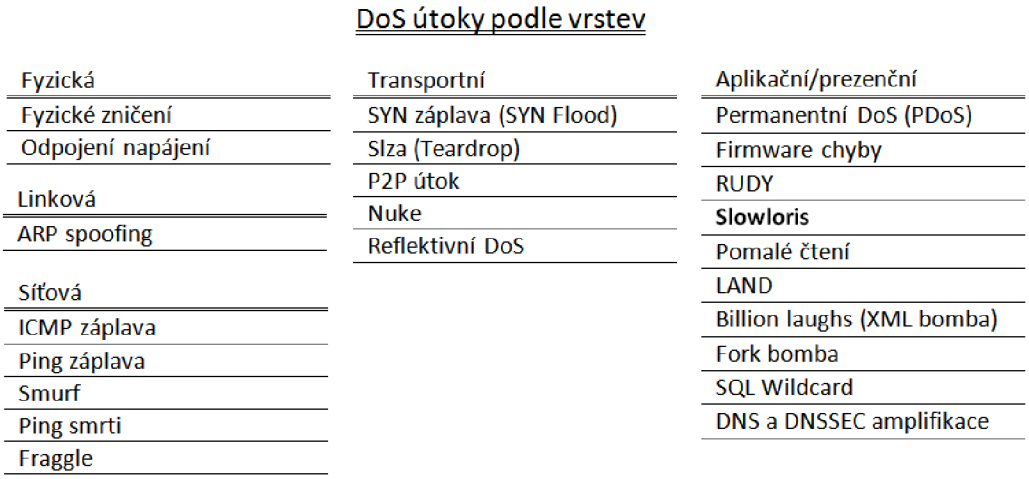
2.2 Základní rozdělení útoků

Útoky lze rozdělit podle mnoha kritérií. Uvádím zde několik základních. Na začátek podle toho, zda útočník je aktivní či pasivní při provádění útoku. Aktivní formy útoků jsou ty, při kterých útočník je aktivně zapojen, například komunikuje se serverem, zasílá na něj data, modifikuje toky přes něj proudící a další. Naopak při pasivním útoku, dochází například pouze k odchytávání dat. Tato forma se často používá před aktivním provedením útoku, pro zjištění stavu sítě, její konfigurace, prostředků a dalších informací, ve kterých potom útočník hledá informace, které používá při přípravě útoku, aby dosáhl lepších výsledků.

Druhé kritérium, podle kterého lze dělit, je cíl či účel útoku. Tím může být například krádež dat, pomsta, osobní uspokojení a další. Velmi zajímavým cílem je zašifrování dat oběti a požadování finanční částky za zpětné dešifrování dat. Dále konkurenční důvody, zničení dat či ovládnutí počítače oběti. Takový počítač, který je v jisté formě ovládán útočníkem, je pro jeho uživatele velmi nebezpečná věc. Nevědomě jej může útočník využívat pro další útoky, lze také sledovat oběť, vyčkávat na její práci například s bankovnictvím, či jinými cennými údaji. Tato forma útoku je v dnešní době poměrně běžná a setkává se s ní v různých formách většina uživatelů.

Posledním kritériem, které zmiňuji, je doba života útoku. Ta většinou určuje i závažnost útoku. Tímto kritériem myslím, jak dlouho už útok existuje, je znám a zda je stále aktivně využíván. Rozložení útoků je vidět na obrázku 2.1, kde v levé straně lze vidět nové hrozby a vpravo ustálené problémy, které jsou stále aktivně využívány. Tuto křivku lze také rozdělit i jako typ chyb. V levé části jde o chyby aplikační, které jsou většinou po krátkém čase napraveny aktualizací softwaru či nějakým balíkem oprav. Proto se jich zde vyskytuje více a s časem velmi řídnou. K využití těchto chyb pro útok často nejsou nástroje a dokáží je využít jen opravdu znalí útočníci, kteří rozumí problému a přesně vědí kam útočit a co použít. Naopak od středu k pravému okraji jsou útoky, které jsou známé již dlouho, ale

[1] Používá se v IT k vyjádření vyspělosti technologie.



Obrázek 2.2: Rozdělení DoS útoků podle vrstev.

stále se vyskytují, protože chyby, které je umožňují, jsou například v návrhu protokolů. K těmto útokům je na internetu spoustu materiálů, návodů a programů, které pomáhají s jejich provedením. A právě do této části patří i DoS útoky.

2.3 DoS útok a jeho modifikace

Podle výše uvedených kritérií lze tedy zařadit DoS[4] útoky mezi aktivní formy útoků, jejichž cílem je vyřadit server, či poskytovanou službu z provozu. Účelem může být poškození konkurence, forma upozornění, osobní důvody, snaha o zviditelnění a další. Také lze říct, že se jedná o ustálený útok, který je starý jako internet sám. Některé důvody, proč použít právě formu DoS útoku jsou popsány zde[1], samotnými uživateli tohoto útoku. Ovšem původní forma DoS útoku by v dnešní době byla těžko proveditelná, ovšem ne nemožná. Proto vznikají různé modifikace zvyšující účinnost. Na obrázku 2.2 jsou zobrazeny některé varianty DoS útoků, které jsou rozděleny podle toho, jakou vrstvu napadají. Bližší popis nejznámějších variant je zde[6].

V dnešní době jsou často DoS útoky tzv. distribuované (Distributed Denial of Service). To znamená, že útok není prováděn z jednoho počítače, ale současně jej provádí více počítačů. K těmto útokům tedy dochází například po předchozí domluvě. Část uživatelů se účastní dobrovolně, často ale větší část nedobrovolně. K tomu dochází pomocí již zmíněného ovládnutí počítačů. Útočník využije virů, přesněji malwarů, pomocí kterých vytvoří síť botů, napadených počítačů, které potom může ovládat. Tuto skupinu pak v daný moment mobilizuje a vyzve k útoku. Takto lze pro provedení útoku využít tisíce počítačů, i když opravdových útočníků, kteří mají zájem na provádění útoku, může být velmi malý počet, například desítky, nebo také pouze jeden. O tak silnou a nebezpečnou variantu se jedná. Tyto útoky se dále liší ve způsobu provedení, to zde není nutné dále vysvětlovat a více lze nalézt například v zajímavém seriále o DoS zde[5].

Kapitola 3

Slowloris útok

Celá tato kapitola je věnována hlavnímu tématu mojí práce a to jedné z méně známých variant Denial of Service útoků a to Slow HTTP DoS[7] útok, také známý jako Slowloris[15], viz sekce 3.2. Hlavní odlišností tohoto útoku od známých DoS a DDoS útoků, které se snaží zahltit linku velkým datovým tokem, je naopak velmi nízký, zato dlouhotrvající tok[12]. A právě tato navenek slabě a tiše působící modifikace je ve své práci velice účinná a nebezpečná. Poté je cíl, který nemá potřebné prostředky k obraně, bezmocný. Jak již bylo naznačeno provést tento útok je velice jednoduché a právě ona jednoduchost a nenápadnost celého útoku je velkým problémem při detekci a obraně proti tomuto napadení.

V první části kapitoly je naznačena základní podstatu celého útoku, dále jsou více rozvedeny použité metody. Jelikož jsem při práci využil i provádění samotného útoku pro testovací účely a k tomu mi sloužil výše uvedený skript, je také stručně popsán v této kapitole. Na závěr je zde vysvětlen důvod neúčinnosti tohoto útoku na některé druhy serverů a uvedeno několik způsobů obrany.

3.1 Princip útoku

Podstata celého útoku je taková, že útočník otevře s cílem velký počet TCP spojení, dle výpočetních kapacit cíle. V některých případech se jedná o stovky spojení, které zahltí server, jindy jsou potřeba tisíce. Tyto spojení jsou navazovány zasláním HTTP GET hlavičky, která je validní, ovšem bez ukončení pomocí řetězce `\r\n\r\n`. Příklad této hlavičky lze vidět na obrázku 3.1. Neukončené hlavičky je dále využíváno pro udržení otevřeného spojení a tím i serveru v nedostupném stavu libovolně dlouhou dobu. Pokud by byla hlavička ukončena, došlo by k uzavření spojení a tím i uvolnění výpočetních prostředků, což by vedlo k neefektivitě útoku. Zde lze nalézt jeden z mála problémů tohoto útoku a to, že dokud se nepodaří obsadit všechna možná spojení, může být server stále pro část uživatelů dostupný. Často ale tento stav netrvá dlouho a všechna spojení jsou v krátké době ovládnuta. Tímto jsou vypotřebovány prostředky serveru pro přijímání nových požadavků a dojde k zahlcení. S otevřením nového spojení, dochází ke spuštění časovače, po jehož vypršení by byl požadavek zahozen a spojení uzavřeno. Nyní už je tedy potřeba jen tyto spojení udržovat aktivní a tím i udržovat server ve stavu, kdy není schopný reagovat na další požadavky. K tomu stačí pouze v určitém časovém intervalu vložit na každé spojení jeden paket s dalším řádkem nekonečné hlavičky, opět bez ukončujícího řetězce `\r\n\r\n`, a tím donutit server resetovat časovač spojení. Jak tato správná, ale neukončená část hlavičky může vypadat je znázorněno na obrázku 3.2. A protože například servery Apache mají výchozí nastavení pro

```
GET / HTTP/1.1\r\n
Host: host\r\n
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1;
            Trident/4.0; .NET CLR 1.1.4322; .NET CLR 2.0.50313;
            .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; MSOffice 12)\r\n
Content-Length: 42\r\n
```

Obrázek 3.1: Příklad neukončené HTTP GET hlavičky, kterou je navázáno spojení.

```
X-a: b\r\n
```

Obrázek 3.2: Příklad další části neukončené hlavičky.

timeout 300 sekund, opravdu 5 minut, stačí jednou za tuto dobu oživit spojení, zasláním další části hlavičky. S minimální námahou tak muže být server udržován v nedostupném stavu hodiny. Velmi jednoduché, velmi účinné a velmi těžko odhalitelné, protože na rozdíl od DDoS útoku je zatížení provozu nízké, disky jsou v klidu a logovací soubory nic nezaznamenávají. Jediným upozorněním je velký počet otevřených spojení.

3.2 Slowloris

Jedná se o skript napsaný v jazyce Perl, s jehož pomocí lze snadno napadnout pomocí Slowloris útoku jakýkoliv server. Tento skript, napsaný Robertem Hansenem, je volně dostupný na domovských stránkách[2] slowlorisu, odkud jej lze bez problémů stáhnout a začít používat. Lze tam také nalézt základní popis a další informace. Celý skript je pak zdokumentován v perldoc. Po několika minutách studia je tedy možné bez problémů začít slowloris.pl používat. Této jednoduchosti jsem využil při vývoji, například pro testovací účely, získání informací pro lepší návrh a dalším.

Na obrázku 3.3 lze vidět přibližnou formu výstupu krátký čas po spuštění. Dále je zde uvedeno několik základních parametrů, se kterými lze skript spouštět a tím měnit jeho konfiguraci.

-dns – Slouží k zadání adresy cíle útoku.

-port – Pomocí tohoto parametru lze změnit cílový port, výchozí hodnota je 80.

-timeout – Tímto se nastavuje interval mezi oživováním spojení v sekundách, výchozí čas je 500s.

-num – Počet otevřených spojení, výchozí počet je 1000.

-test – Pro spuštění testu nastavení časovače cíle, viz obrázek 3.4.

3.3 Kdy útok nefunguje

Různé servery mají různé způsoby přijímaní požadavků a jejich následné zpracování. Tento fakt způsobuje, že například na serverech IIS, lighttdp a několika dalších je útok méně

```
./slowloris.pl -dns localhost -port 80 -timeout 500 -num 500
Welcome to Slowloris - the low bandwidth, yet greedy and poisonous HTTP client
Multithreading enabled.
Connecting to localhost every 500 seconds with 500 sockets:
                Building sockets.
                Building sockets.
                Building sockets.
                Building sockets.
                Building sockets.
                Building sockets.
                Sending data.
Current stats: Slowloris has now sent 468 packets successfully.
This thread now sleeping for 500 seconds...

                Sending data.
Current stats: Slowloris has now sent 538 packets successfully.
This thread now sleeping for 500 seconds...

                Building sockets.
                Building sockets.
                Sending data.
Current stats: Slowloris has now sent 772 packets successfully.
This thread now sleeping for 500 seconds...
```

Obrázek 3.3: Příklad výstupu skriptu slowloris.pl při spuštění útoku.

účinný až neúčinný. Seznam serverů, na které by Slowloris útok měl a neměl fungovat je na domovských stránkách Slowlorisu[2], kde jsou označeny i servery na kterých bylo provedeno testování. Dalším zdrojem informací co se testování serverů může být[13].

Důvodem neúčinnosti je například různé chápání práce s časovačem, nebo různé zpracování příchozích požadavků. U ohrožených serverů Apache je s otevřením spojení spuštěn časovač a k jeho resetování dochází po přijmutí paketu. A protože je navázáno i vlákno pro další zpracování požadavku, i když je v podstatě bez využití, dokud není požadavek kompletní, tak si server drží prostředky pro potřebné výpočty při zpracování požadavku. A jelikož jsou při útoku využity právě nekompletní požadavky, lze jej pro útok na tyto servery efektivně použít. Naopak jiné servery pracují s časovačem tím způsobem, že při otevření spojení jej spustí a do jeho vypršení přijímají požadavek. Pokud do té doby není kompletní, je zahozen a spojení uzavřeno. Dále například IIS servery využívají jistou formu předzpracování požadavků, která nezabírá výpočetní kapacity serveru. Tudíž na ně není útok, prováděný formou otevření velkého počtu spojení a jejich udržování se v aktivním stavu, účinný.

3.4 Možnosti obrany

Výše jsem nastínil, jak útok funguje a proč je někdy naopak neúčinný. Z toho vycházejí způsoby, jak se lze bránit, i když ani ty nejsou úplně spolehlivé, ale výrazně snižují efektivitu útoku. Podle[10] jsou čtyři základní možnosti, které jsou popsány níže. Jistě existují i jejich další modifikace, ale ty není nutné dále rozebírat. Podrobnější popis způsobů ochrany přes útok lze nalézt zde[14].

```
./slowloris.pl -dns localhost -port 80 -test
Defaulting to a 5 second tcp connection timeout.
Multithreading enabled.
This test could take up to 14.3666666666667 minutes.
Connection successful, now comes the waiting game...
Trying a 2 secons delay:
            Worked.
Trying a 30 secons delay:
            Worked.
Trying a 90 secons delay:
            Worked.
Trying a 240 secons delay:
            Worked.
Trying a 500 secons delay:
            Worked.
Okay that?s enough time. Slowloris closed the socket.
Use 500 seconds for -timeout.
```

Obrázek 3.4: Výstup skriptu slowloris.pl při spuštění testování časovače.

Jedním z nich je úprava konfigurace serveru. Jak jsem uvedl výše, nastavení serverů Apache na timeout 5 minut je v dnešní době rychlých připojení pouze natažená ruka právě útočníkům. Takže lze upravit tuto hodnotu, například na 30 sekund. Ovšem i s tímto skript pro provedení útoku Slowloris počítá, takže nejen, že lze zvolit čas, po kterém dojde k obnovení spojení, ale lze před samotným útokem spustit test, který vytvoří několik spojení s cílem a po daných intervalech tyto spojení zkontroluje, zda jsou stále aktivní, viz obrázek 3.4. Tímto lehce útočník zjistí jaký maximální timeout si může pro útok nastavit. Ač je tato úprava konfigurace sama o sobě méně účinná, v kombinaci s dalšími opatřeními dokáže útočníkovi velmi znepříjemnit práci, případně úplně zabránit úspěšnému provedení útoku.

Další možností je upravit maximální počet spojení z jedné IP adresy. Tato změna už výrazně zvyšuje šanci ubránit se útoku. Ale zde je potřeba brát v úvahu více věcí, abychom například špatnou konfigurací neblokovali i regulérní požadavky. Problémem může být například firma, škola či jiná větší organizace využívající NAT. Potom by za určité situace mohli být požadavky z těchto míst zablokovány, i když k žádnému napadení nedochází.

Třetí možností je připojení modulu, který bude hlídat příchozí požadavky. Tímto lze velmi zmírnit účinnost Slow HTTP DoS útoků. Několik těchto modulů je volně dostupných na internetu. Například mod_antiloris[9] počítá spojení z IP ve stavu čtení požadavku a pokud toto číslo překročí nastavený limit, další již nejsou přijímána. Tím lze spolehlivě ochránit server před napadením.

Je tu ještě jeden způsob, zřejmě nejlepší a to připojení hardwarového zařízení, které bude předzpracovávat příchozí HTTP požadavky podobně jako IIS servery a serveru dál předá pouze ty kompletní, které mohou být ihned obslouženy a nebrání dalším výpočetním úkonům. Ovšem tato varianta bude zřejmě nejvíce nákladná a její použití tedy není až tak časté.

Kapitola 4

Nemea

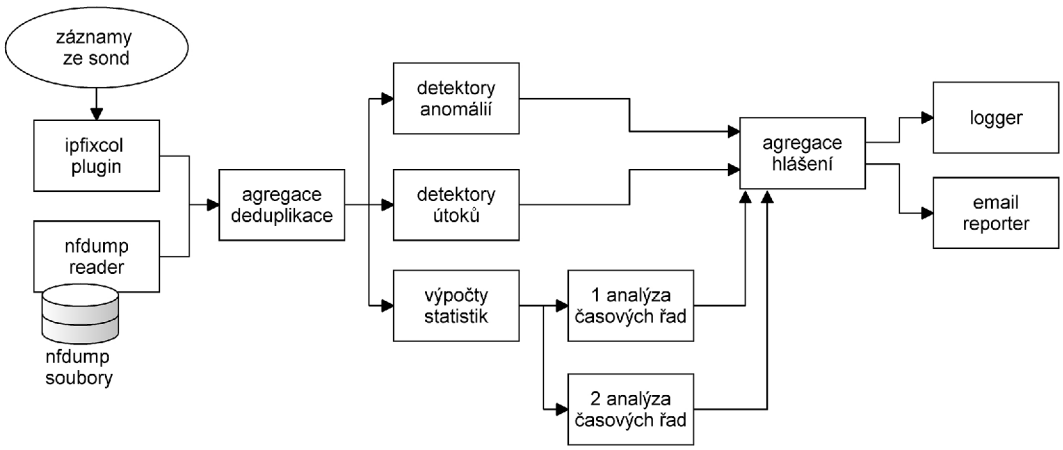
Jelikož útoků na počítačové sítě je mnoho a objevuje se stále více typů, není snadné je odhalit natož se jim bránit. A i když je většina již známá a existují informace, čím se jednotlivé útoky vyznačují, jaké jsou jejich parametry, kde mají slabé stránky a mnoho dalších, pokrytí nejčastěji používaných útoků je v běžných podmínkách velmi těžké. Z těchto důvodů vznikají systémy, které monitorují běh sítě a jejich úkolem je detekce anomálií, útoků, výpadků a dalších problémů. Použití takovýchto systémů v praxi poté vede ke snížení počtu úspěšných útoků, odstranění problémů a k celkovému zlepšení kvality komunikace.

Jedním z těchto systémů je i projekt Nemea[3]. Zmiňuji se zde o něm, protože cílem mé práce je vytvořit detekční modul právě pro tento systém. V dalších odstavcích lze tedy nalézt základní informace týkající se tohoto projektu. Konkrétněji v první části jde o popis a přiblížení celého systému. Dále se text zabývá moduly, jež v podstatě tvoří celý systém. V závěru je popsáno rozhraní TRAP, které je vytvořeno pro jednotnou komunikaci modulů a formát UniRec sloužící k přenášení dat a snazšímu přístupu.

4.1 Co je Nemea

Network Measurements Analysis, zkráceně Nemea, je projekt zabývající se vytvořením systému pro analýzu síťových dat, především pomocí záznamů o tocích. Základem celého projektu je jednoduchost a modulárnost, čehož je dosaženo rozdělením systému na jednotlivé moduly komunikující ve specifikovaném formátu přes vytvořené rozhraní. Spojením jednotlivých modulů, provádějící konkrétní operace potom získáme systém, který například přečte data, zpracuje je a vyhledá problém. Takto lze jednoduše vytvořit efektivní, flexibilní a lehce distribuované systémy na analýzu síťového provozu.

Celý systém je rozdělen na několik hlavních částí, které se mohou lišit podle modulů, jež jsou zapojeny. Jedna z variant je popsána dále a je vidět na obrázku 4.1. První část slouží pro získávání dat, ošetření a jejich následné zpracování do formátu pro snazší práci s nimi v dalších modulech. Tyto první moduly umí také data například ukládat pro testovací účely, tím vznikají tak zvaná offline data, která jsou ve formátu nfdump souborů. Dále tato data procházejí, přes střední část, kde jsou různé statistické moduly, detektory útoků a anomálií. Do této části je směřován i výsledek mojí práce. Výstupy z těchto modulů poté pokračují do části, kde probíhá zpracování výsledků z detektorů a dalších modulů. Po vyhodnocení záznamů jsou vytvořena hlášení, logovací zprávy či zpětná vazba k datům. Navíc jsou zde další moduly, které mají vedlejší využití například pro testovací účely. Mezi tyto moduly patří například ukázkové moduly, generátory útoků, modul pro čtení offline dat a další.



Obrázek 4.1: Názorné rozdělení systému Nemea.

Všechno výše uvedené vede k tomu, že i když jde o větší a stále rostoucí systém, není problém se v něm začít orientovat v krátké době, lehce jej rozšiřovat či jinak modifikovat. K pochopení celého systému a bezproblémové práci stačí krátké studium, ke kterému je dostatek podkladů a také velmi ochotný přístup lidí pracujících na tomto projektu je nápomocný.

4.2 Moduly

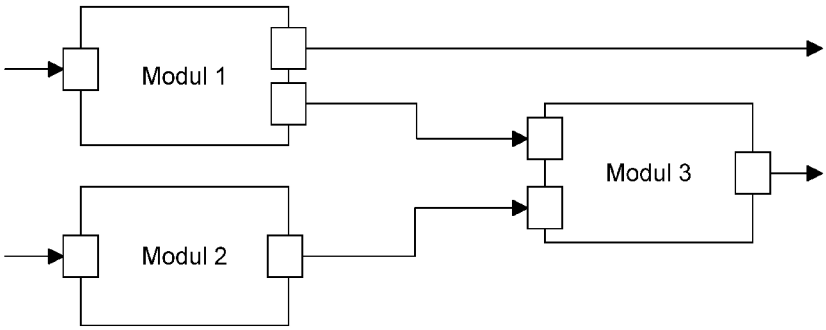
Jak bylo zmíněno, systém je složen z modulů. Většina z nich je napsaná v jazyce C nebo C++, ale najdou se i výjimky v Pythonu. Každý z těchto modulů běží jako samostatný program, využívající knihovnu TRAP, která poskytuje rozhraní. Modul může mít 0-N vstupních rozhraní, na kterých přijímá data, která dále zpracovává, počítá a poté na 0-N výstupních rozhraní zapisuje výsledky své práce. Přes rozhraní proudí moduly libovolná data, uložená ve formátu UniRec, což velmi usnadňuje práci s nimi, více je popsáno níže.

Každý modul má jistou strukturu, kterou je nutné dodržovat, což má jisté výhody, například velmi snadné vytvoření nového kompatibilního modulu. Po dodržení těchto základních pravidel není problém během krátkého času uvést do běhu základní modul a tím se více soustředit na řešení samotného problému. Další výhodou je velká přehlednost zdrojových kódů, jelikož jsou psány v jistém, skoro neměnném stylu.

4.3 Rozhraní modulů TRAP

Pro práci s rozhraními, využívají moduly TRaffic Analysis Platform (TRAP), jejímž základem je dynamická knihovna libtrap, poskytující jednotné rozhraní pro komunikaci modulů. Velmi důležitou a zajímavou funkcí TRAP je, že při chybách nedochází k hlášení chyb, ale automatickému zotavení. Na obrázku 4.2 lze vidět příklad propojení modulů pomocí rozhraní TRAP.

Jak je napsáno výše, každý modul může komunikovat přes 0-N rozhraní, ať vstupních či výstupních. Jejich počet nemusí být shodný. Specifikace počtu vstupních a výstupních rozhraní je uvedena v modulu. To slouží ke kontrole, protože rozhraní se nastavují při spuštění modulu pomocí parametrů. Těmi je určeno o jaký typ rozhraní se jedná a kam se



Obrázek 4.2: Rozhraní modulů poskytující TRAP.

má připojit. Potom je důležité, aby vzájemně komunikující moduly měly shodné rozhraní, přes které komunikují, jinak se nezdaří připojení.

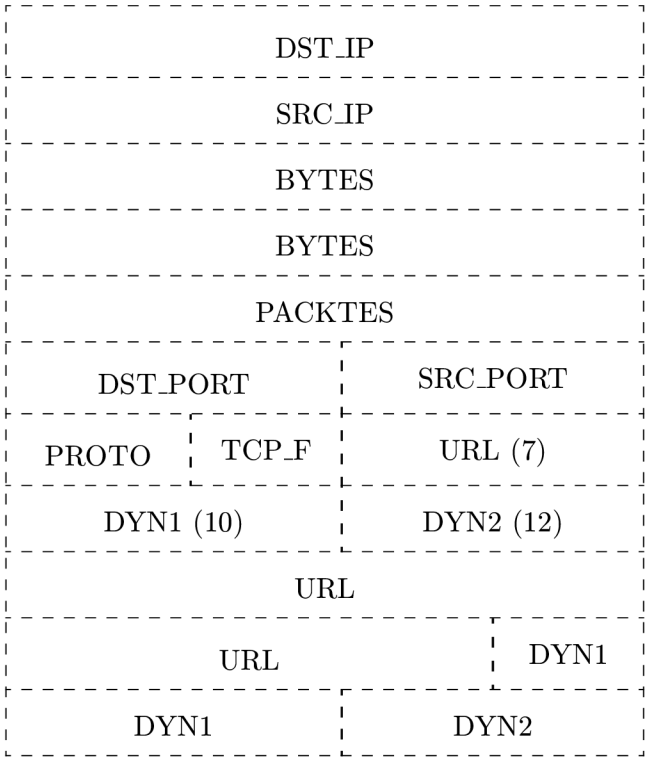
Jelikož tato knihovna také zajišťuje několik základních funkcí každého modulu, velmi to usnadňuje implementaci. Protože se tyto funkce často shodují pro všechny moduly, existují zde makra, která pak stačí použít při psaní programu a tím se vyhnout vedlejším problémům. Těmito makry lze vyřešit inicializaci modulu, zpracování parametrů, zachytávání signálů, čtení a zápis dat a také ukončení modulu, což mi přijde velice užitečné. Lze tak rychle vytvořit funkční modul a přitom se stále soustředit na hlavní funkci modulu.

4.4 Formát UniRec

UniRec je datový formát, který se využívá v systému Nemea. Je to forma záznamu dat, například o toku, která obsahuje položky. Vznikají zde proto šablony, které jsou uspořádáním položek do určité podoby, se kterou lze dále pracovat. V systému existuje několik základních šablon, například pro záznam o toku, ale není problémem vytvořit novou šablonu. Položky v šabloně mohou být statické či dynamické, což znamená proměnlivé velikosti. Vždy jsou první uložené statické a za nimi dynamické, přičemž ve statické části jsou údaje o velikosti dynamických částí. Na obrázku 4.3 je znázorněna šablona záznamu toku s dynamickými položkami.

Na základě šablony je potom vytvořena struktura pro uložení dat, do níž jsou zapisována data ze vstupního rozhraní. Ta jsou dále zpracována, a pokud modul vytváří výstupy, zapíše data do výstupní struktury, přes kterou jsou předány výstupnímu rozhraní. Nejen pro všechny tyto operace jsou zde vytvořené funkce, ale dále také například pro zápis a čtení statických i dynamických položek, zjišťování velikosti položek, či vytváření a uvolňování šablon.

Jelikož jsou v UniRec uloženy i záznamy o tocích, které jsou z formátu IPFIX převáděny, velice to usnadňuje a urychluje další práci s daty. Nemusí totiž docházet k parsování, ale pomocí makra `ur_get` pro přístup k datům a `ur_set` pro zápis. Vše se tím stává mnohem přehlednější.



Obrázek 4.3: Šablona UniRec záznamu.

Kapitola 5

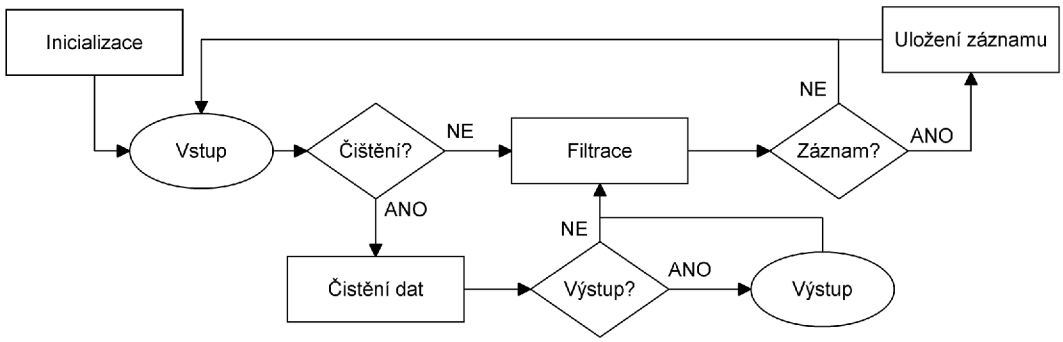
Návrh modulu pro detekci Slowloris útoků

Jelikož hlavním cílem mé práce je vytvořit modul, který bude součástí systému složeného z několika dalších, podobně pracujících modulů, bylo základem dobrého návrhu prostudování systému a okolních modulů. Tím jsem snadno získal poměrně jasnou představu o základní struktuře práce. Tato část návrhu je popsána v první části této kapitoly. Následována popisem návrhu dat, která jsou potřebné pro správnou práci modulu. Jelikož na začátku návrhu jsem pro nedostatek informací nemohl rozhodnout všechny problémy, byly další fáze návrhu prováděny průběžně s implementací, na základě získaných poznatků.

Jak celkový návrh vypadá je vidět na obrázku 5.1, kde je zobrazeno několik funkčních částí modulu. Popis několika částí modulu lze nalézt v této kapitole. Jednou z částí kapitoly je popis vstupu a výstupu, potom také podrobně popsán návrh filtrace. Dále je zde také rozebráno, jakým způsobem jsou označovány záznamy pro usnadnění práce s nimi. Problematika implementace, inicializace, čištění záznamů a další detaily jsou popsány dále v kapitole. 6.

5.1 Základní struktura modulu

Základem celého návrhu, bylo prozkoumání systému Nemea a některých modulů. Nejvíce přitom byly zkoumány moduly pracující na stejném způsobu jako navrhovaný, to znamená, přijímají záznamy o tocích, zpracovávají je a podle potřeby vytváří odpovídající výstupy.



Obrázek 5.1: Návrh modulu.

Tím byla tato část návrhu poměrně jasně dána a to také vedlo k dodržení jednotné struktury modulů, což velmi zvyšuje přehlednost celého systému. Z tohoto důvodu nebyla vytvářena odlišná stavba modulu.

5.2 Návrh datové části

Jelikož je vytvářen detekční modul a základem správné detekce je vždy dostatek dat, na jejichž základě se rozhoduje, je nutné dobře navrhnout, jaká data budou potřeba a která by jen zbytečně zatěžovala. Také je potřeba myslet na případné výstupy, které musí nést dostatek informací o detekci. Proto jsem zvolil následující položky[1].

Zdrojová a cílová IP adresa – Jelikož významným prvkem v detekci Slowloris je počet spojení mezi dvěma adresami (`SRC_IP`, `DST_IP`), je nutné si tyto informace o komunikaci uchovat. Tato kombinace bude také jednoznačným identifikátorem každé položky o komunikaci, která bude uložena pro další potřeby.

Čas prvního uloženého záznamu – Časová informace (`TIME_LAST`) o vytvoření položky komunikace v poli, aktuální čas při uložení prvního záznamu. Bude použita při vytváření výstupů, jako upřesňující informace času detekce začátku útoku.

Čas posledního uloženého záznamu – Časová informace o zpracování posledního záznamu dané položky komunikace v poli (`TIME_LAST`). Tato informace bude v položce příslušné komunikace měněna při každém příchodu odpovídajícího záznamu a bude sloužit k detekci a při vytváření výstupů.

Počítadla přenesených bajtů a paketů – Informace o počtu bajtů (`BYTES`) a paketů (`PACKETS`), přenesených během komunikace mezi dvěma adresami. Při příchodu nového záznamu budou počítadla v odpovídající položce zvýšena o příslušnou hodnotu. Tyto informace budou sloužit při vytváření výstupů.

Počet záznamů pro danou komunikaci – Počítadlo záznamů, uložených u komunikace dvou adres, které je hlavním klíčem k detekci útoku. Hodnota, která bude sloužit k označení komunikace za útok, je popsána společně s dalšími informacemi v sekci 5.5.

Označení podezřelé komunikace – Každá položka o komunikaci ponese označení hodnotou 1-5, její význam lze nalézt v tabulce. Další informace o způsobu označování záznamu jsou popsány v sekci 5.5.

Počítadlo čistících cyklů bez příchodu nového záznamu pro danou položku – Tato informace bude sloužit pro účely odstraňování dat o podezřelé komunikaci, která již dále neprobíhá. Bližší popis použití je uveden v sekci 5.5.

Pro uložení těchto informací o jedné komunikaci bude potřeba vytvořit odpovídající datový formát. A jelikož počet těchto položek, které budou s příchodem záznamů o nových komunikacích přibývat a při čištění ubývat, bude k uložení použito formátu s proměnlivou délkou. Jaké řešení je zvoleno a jeho popis lze nalézt v kapitole 6.

[1] V závorkách jsou uvedeny názvy položek používané v systému Nemea, které se objevují v dalším textu práce.

5.3 Vstup a výstup

Protože v systému Nemea slouží k přenosu dat formát UniRec, bylo zde nutné vytvořit šablony pro vstup a výstup. Tyto šablony budou při implementaci použity pro vytvoření UniRec struktur, kde vstupní bude přijímána přes vstupní rozhraní a výstupní struktura bude naopak při hlášení, předávána pomocí výstupního rozhraní dále.

Volba výstupní šablony byla provedena na základě výstupního formátu modulů, které interpretují online či offline data, aby byla zajištěna bezchybná komunikace. Jak je popsáno výše v sekci 5.2, z této šablony budou důležitými položkami pro uložení `SRC_IP, DST_IP, PACKETS, BYTES, TIME_LAST`. Další položky jsou využity při filtrování příchozích záznamů, což je podrobněji popsáno v sekci 6.5. Těmi jsou `TIME_FIRST, TCP_F, PROTO, DST_PORT`.

Výstupní šablona je vytvořena podle informací, které jsou popsány výše v sekci 5.2. Nyní obsahuje položky `SRC_IP, DST_IP, PACKETS, BYTES, DETECTION_TIME, TIME_LAST`. Její formát bude v budoucnu zřejmě upřesněn podle požadavků modulu, který bude zpracovávat hlášení detektorů. Proto zde lze prozatím vidět základní informace o útoku jako zdrojovou adresu a cílovou adresu, čas detekce útoku a čas ukončení útoku, a počet přenesených bajtů a paketů.

5.4 Návrh filtrace

Proces filtrace je jednou z nejdůležitějších částí celého modulu. Zde totiž bude docházet k rozpoznávání podezřelých toků od běžných. A jelikož je detekce založena na počtu toků během komunikace mezi dvěma adresami, je velmi důležité, aby filtrací procházelo co nejméně záznamů z normálního provozu, ale naopak, co nejvíce záznamů příchozích při útoku.

Při vytváření spolehlivého filtru je proto nutné zvolit správný seznam parametrů a k tomu dále sadu hodnot, která bude správně filtrovat požadované záznamy. Volba parametrů v mém případě byla otázkou studia útoku. Naopak pro získání správné sady hodnot, bylo potřeby provést několik desítek analýz. Popis jednotlivých parametrů a vyhodnocení analýz pro jednotlivé parametry lze nalézt dále v této části.

Jako první jsou popsány parametry, jejichž hodnoty byly jasné již z podstaty útoku. Jedná se protokol spojení, které musí být spojové, to znamená TCP. Tento parametr nese položka `PROTO` a hodnota, která označuje TCP je 6. Tímto parametrem dochází k odfiltrování více jak poloviny toků. Následuje testování na cílový port, jehož číslo je v položce `DST_PORT`. A protože tento útok je směřován na webové služby, je zvolena hodnota 80.

K dalším parametrům již byla potřebná analýza záznamů, kterou jsem prováděl pomocí skriptu slowloris.pl, viz 3.2, který provádí útok. Spuštěním několika desítek útoků s různými možnostmi nastavení a zpracováním příslušných toků, které jsem pomocí zdrojové a cílové adresy mohl vyfiltrovat, jsem získal přehled o četnosti výskytu jednotlivých hodnot. Z toho jsem dále určoval jaké hodnoty u každého parametru zvolit.

První získaný parametr je délka času záznamu. Ta je počítána jako rozdíl mezi položkami `TIME_LAST` a `TIME_FIRST`. Výsledky pro tento parametr jsou uvedeny v tabulce 5.1 a je jisté, že hodnotou bude 0. A jelikož jsou výsledky jednoznačné, je tento parametr kontrolován ihned po neměnných.

Dalším parametrem je počet paketů, protože se zde také výsledky poměrně málo rozchází. Pro kontrolu je zvolen interval 1-4 paketů, který se nejčastěji vyskytoval. Další hodnoty položky `PACKETS` a jejich četnost lze nalézt v tabulce 5.2.

Čas v s	% výskytu
0-0	100%
1-5	0%
6-10	0%
≥ 11	0%

Tabulka 5.1: Četnost délek toků při Slowloris útoku.

Počet paketů	% výskytu
1-4	90%
5-10	2%
11-15	3%
≥ 16	5%

Tabulka 5.2: Četnost výskytu počtu paketů při Slowloris útoku.

Velmi důležitým parametrem objeveným během analýz je průměrný počet bajtů v jednom paketu, počítaný jako poměr položek `BYTES` a `PACKETS`. Jelikož samotné hodnoty počtu bajtů jsou rozloženy vcelku rovnoměrně, viz tabulka 5.3, bylo potřeba najít náhradu za tento parametr. Jak je vidět v tabulce 5.4, která ukazuje četnost jednotlivých intervalů hodnot bajtů na paket, je zde větší jednoznačnost, díky které lze tento parametr použít.

Posledním zvoleným parametrem je `TCP_F`, což jsou příznaky TCP komunikace. Zde je zvolen trochu jiný postup, protože se vykytuje větší počet hodnot, které tato položka může nabývat. Proto byly v první řadě získány tyto hodnoty a potom zjišťována jejich četnost. Výsledky jsou v tabulce 5.5 a jsou velmi rovnoměrné, což znamená, že ani jedna hodnota toky útoku výrazně nespecifikuje, ale protože se ve velké míře objevují pouze tyto hodnoty, jejich použití v kombinaci s výše uvedenými zvýší účinnost filtru. Proto byly kromě hodnoty 27 použity.

Pro shrnutí zde na obrázku 5.2 uvádím diagram celé filtrační části, kde lze přehledněji vidět všechny parametry a hodnoty výše popsané.

5.5 Označování záznamů

Pro lepší práci se záznamy bude vhodné je rozdělit od několika kategorií, podle kterých modul bude lépe rozhodovat jak se záznamem dále pracovat. Toto označení je provedeno jednoduše číslem v intervalu 1-5, které bude mít každá položka uložené, jak je uvedeno výše v sekci 5.2. Význam jednotlivých hodnot je krátce vysvětlen v tabulce 5.6 a zde je více rozebrán.

Počet bajtů	% výskytu
0-100	39%
101-300	13%
301-500	25%
501-1000	12%
≥ 1001	11%

Tabulka 5.3: Četnost výskytu počtu bajtů při Slowloris útoku.

Bajtů/paketů	% výskytu
0-39	0%
40-50	61%
51-99	4%
100-150	31%
$\geq$ 151	2%

Tabulka 5.4: Četnost výskytu počtu bajtů na paket při Slowloris útoku.

Příznak	% výskytu
2	12%
16	13%
17	8%
24	22%
25	16%
26	23%
27	3%
ostatní	3%

Tabulka 5.5: Četnost výskytu příznaků při Slowloris útoku.

1 – Tímto číslem je označena každá položka, jejíž záznam prošel filtrací a byl uložen do paměti modulu. Při čištění, viz sekce 6.4, jsou tyto položky odstraněny.

2 – Toto číslo označuje položky, u kterých je uloženo více jak 150 záznamů a tudíž se tato položka stává podezřelou. Takto označené položky jsou odstraňovány na základě počítadla čistících cyklů. Jeho hodnota je vždy při čištění zvyšována o 1, ale při příchodu nového záznamu dané položky je toto počítadlo nulováno. K odstranění dochází při dosažení hodnoty počítadla 10, což je přibližně 5 minut. Ovšem je zde nutné upozornit, že modul při práci nevyužívá reálného času, viz sekce 6.3.

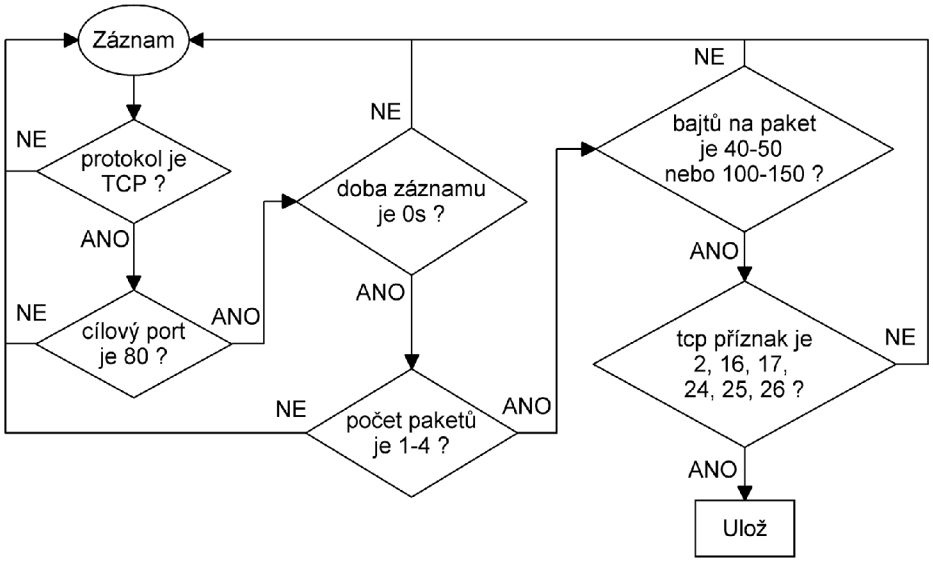
3 – Takto jsou označeny položky vyhodnocené jako útok. K tomu dochází při překročení 2000 uložených záznamů u dané položky. Po vytvoření výpisu o tomto útoku, je tato položka označena číslem 4.

4 – Položky označené touto hodnotou jsou probíhající útoky, o kterých se dále sbírají informace, proto nejsou odstraňovány. Při čištění se kontroluje čas poslední aktualizace položky, pokud tento čas překročí 5 minut, je záznam označen hodnotou 5.

5 – Takto značná položka znamená ukončený útok, o kterém je vytvořeno druhé hlášení,

Úroveň	Popis	Akce při čištění záznamů
1	Normální záznam	Smazání
2	Podezřelý záznam	Ponechání pro získání dalších informací
3	Detekován útok	Vytvoření zprávy o probíhajícím útoku
4	Ohlášený útok	Ponechání do doby ukončení útoku
5	Ukončený útok	Vytvoření zprávy o celém útoku

Tabulka 5.6: Hodnoty pro označení záznamů a jejich popis.



Obrázek 5.2: Diagram filtrační části modulu.

obsahující informace nasbírané za celý průběh útoku. Po vytvoření výstupu je položka označena číslem 1, aby došlo k jejímu následnému smazání.

Kapitola 6

Implementace modulu

Vývoj modulu mohl být prováděn na serveru Benefizio, což je vývojový server celého projektu Nemea, nebo po vytvoření lokální kopie systému Nemea na vlastním stroji. Jelikož mi druhá varianta přišla více pohodlná, rozhodl jsem se pro ni. Vývoj tedy probíhal pod 64bitovým operačním systémem Ubuntu verze 13.10. Modul je napsán v jazyce C, protože v tomto jazyce je vytvořena podstatná část systému Nemea a tudíž byl i zadán.

Dále v této kapitole jsou popsány způsoby implementace jednotlivých částí modulu, které jsou znázorněny v diagramu na obrázku 5.1. V první části se věnuji inicializaci, po níž následuje popsání implementace vstupu a výstupu. Dále vysvětlení práce s časem a způsob čištění dat. Předposlední část se věnuje nejdůležitějšímu problému a to filtraci a ukládání záznamů. Na závěr krátké zhodnocení celé fáze implementace.

6.1 Inicializace

Po spuštění modulu proběhne inicializace, při které se vytvoří všechny věci potřebné pro další běh modulu. Ta probíhá ve čtyřech krocích, které jsou popsány dále.

V první části jsou pomocí makra `TRAP_DEFAULT_INITIALIZATION`, které je součástí knihovny celého systému, vytvořena a připojena rozhraní. Nastavení těchto rozhraní je předáno pomocí parametrů při spuštění modulu, proto zde také proběhne kontrola správnosti nastavení.

Dalším krokem je vytvoření UniRec šablon na základě návrhu, které jsou popsány v sekci 5.3. K této operaci rovněž slouží funkce `ur_create_template`. Pro vytvoření výstupní struktury, která bude předávána na výstupní rozhraní, je nutné ještě zavolat funkci `ur_create`.

Dále je nutné vytvořit datové struktury pro ukládání příchozích záznamů. K uložení jednoho záznamu je vytvořen datový typ `flow_record_t`, jenž je strukturou, která je zobrazena na obrázku 6.1. Jednotlivé položky této struktury jsou popsány výše v sekci 5.2 a bližší popis jejich využití po implementační stránce je dále v textu. A protože jak již bylo uvedeno, počet těchto položek se bude neustále měnit, je pro jejich uložení vytvořeno pole `flow_records`, které je proměnné délky a jeho výchozí velikost je 4096.

V poslední části dojde k inicializaci proměnných, které slouží například pro práci s časem, počítadlem uložených položek a další.

Jelikož je modul určen k běhu bez ustání a v reálném čase, je tak i formován, kdy po inicializaci se dostane do nekonečné smyčky a k jeho správnému ukončení slouží zaslání signálu SIGTERM nebo SIGINT.

```
typedef struct {
         ip_addr_t src_ip;
         ip_addr_t dst_ip;
         ur_time_t detection_time;
         ur_time_t end_time;
         uint32_t packets;
         uint64_t bytes;
         int flow_counts;
         uint8_t level;
         uint8_t clean_count;
} flow_record_t;
```

Obrázek 6.1: Struktura slouží k uložení informací o toku.

6.2 Implementace vstupu a výstupu

Jelikož moduly v systému Nemea využívají pro svoje vstupy a výstupy jednotné rozhraní, existují k němu i funkce, které z něj čtou nebo do něj zapisují. Proto implementace těchto prvků programu nebyla nijak náročná a šlo spíš o pochopení práce s těmito funkcemi a jejich správné použití.

Pro čtení ze vstupního rozhraní tedy slouží funkce `trap_get_data`, které uloží data. Tato operace je ošetřena návratovou hodnotou, která je dále zkontrolována makrem `TRAP_DEFAULT_GET_DATA_ERROR_HANDLING`. Také přijatá data jsou zkontrolována, zda odpovídají UniRec šabloně pro vstup.

Naopak při výstupu je první naplněna výstupní šablona odpovídajícími daty a ta je pak pomocí struktury, vytvořené při inicializaci, předána funkcí `trap_send_data` výstupnímu rozhraní. Opět po této operaci dochází ke kontrole návratové hodnoty makrem `TRAP_DEFAULT_SEND_DATA_ERROR_HANDLING`.

6.3 Práce s časem

Jak bylo zmíněno výše, modul pro svoji práci nevyužívá reálného času, protože se práce s časem velmi liší při zpracování offline a online dat. Z tohoto důvodu bylo nutné naimplementovat řešení, které tento problém odstraní. K tomu velmi napomáhají časové značky `TIME_LAST`, jimiž jsou označeny příchozí záznamy.

Každý záznam nese označení, kdy byl ukončen a jelikož jsou záznamy poměrně dobře seřazeny za sebou, je možné jejich značky využít jako informaci o aktuálním času. Tudíž je v modulu vytvořena proměnná `current_time`, do které je vždy po načtení záznamu uložena hodnota `TIME_LAST`, převedená na sekundy.

Takovéto řešení je použitelné při zpracovávání obou druhů dat, bez nutnosti rozlišovat o který druh se jedná a uzpůsobovat se práci s ním. Jediným drobným problémem je, že může přijít záznam se starším časovým razítkem, jakoby v minulosti, anebo naopak s vyšší hodnotou, který posune hodnotu aktuálního času do budoucnosti oproti ostatním příchozím záznamům. Na to je nutné pomýšlet, při práci s časem, například při zjišťování doby mezi záznamy, musí být ohlídáno, zda později příchozí záznam má i pozdější nebo shodný čas s druhým kontrolovaným.

6.4 Čištění dat

Objem dat, proudící přes modul bude velký a proto musí být vytvořeny operace, které budou množství dat redukovat. Těmto operacím slouží označení položek, které bylo vysvětleno v sekci 5.5. Samotná fáze čištění pak probíhá vždy po více než 30 sekundách. Jak je ale uvedeno výše, nejde o 30 sekund reálného času, nýbrž o rozdíl mezi příchozím záznamem v době posledního čištění a aktuálně příchozího. To znamená, že v reálném čase bude k čištění docházet v určitém intervalu, který by měl mít střed ve 30 sekundách.

V modulu je tedy k provádění čistícího cyklu přistoupeno tehdy, když rozdíl hodnot `current_time` a `last_clean_time`, dosáhne čísla 30 a více. Čištění je prováděno procházením položek pomocí `for` cyklu. Při průchodu je kontrolováno označení `level`, zda nemá hodnotu 1, případně 2 v kombinaci s hodnotou počítadla čištění `clean_count` menší než 10. Pokud je taková položka nalezena, je přesunuta na začátek pole nebo na první volnou pozici, za takto řazenými položkami. Dále je zvýšeno počítadlo přesunutých záznamů, jehož hodnota je po ukončení čištění uložena jako aktuální počet položek uložených v poli, proměnná `cnt_flows`. Při dalším běhu modulu, jsou záznamy určené k smazání přepisovány nově ukládanými položkami.

Jelikož četnost výskytu podezřelých položek je nízká a často je pole vymazáno úplně, probíhá na závěr kontrola počtu zbylých položek vzhledem k výchozí velikosti pole, protože v průběhu ukládání nových položek může docházet ke zvětšování pole, ale o tom je více napsáno dále. Pokud tedy lze, je pole zmenšeno na výchozí velikost, a tím i vyřešen problém práce s pamětí. Dále je uložen aktuální čas jako čas posledního čištění. Tím je fáze čištění ukončena a modul pokračuje v detekční činnosti.

6.5 Filtrování a ukládání dat

Po načtení příchozího záznamu dochází ke kontrole, zda jeho parametry odpovídají filtračním pravidlům. O těch je již napsáno dostatek, viz sekce 5.4, proto je zde více nebudu popisovat, co se parametrů a hodnot v těchto pravidlech týče. Tyto filtrační pravidla jsou v modulu provedeny pomocí do sebe zanořených podmínek `if`. Toto řešení je zvoleno z důvodu velkého množství záznamů, které proudí modulem a režie při načítáních všech dat u každého by byla zbytečná zátěž. Tudíž mezi jednotlivými podmínkami dochází pouze k načítání dat potřebných k vyhodnocení další podmínky.

Záznam, který splní všechny podmínky filtrace, bude uložen. Nejprve je ale nutné zjistit, zda už existuje v poli položka o komunikaci, ke které daný záznam připadá, anebo je nutné vytvořit novou položku. Protože jednoznačným identifikátorem každé položky je kombinace zdrojové a cílové adresy, jsou tyto položky načteny ze záznamu. Potom se cyklem projde celé pole a pomocí makra pro porovnávání dvou IP adres `ip_cmp` se hledá odpovídající položka.

Je-li nalezena, přepíše se čas posledního záznamu `end_time`, o příslušné hodnoty se zvýší počítadla bajtů a paketů, je zvýšeno počítadlo záznamů `flow_count` a vynulováno počítadlo čistících cyklů `clean_count`.

Po úpravě položky proběhne kontrola, zda její počet záznamů `flow_count` nepřesáhl hodnotu 2000, což by znamenalo detekci útoku a zvýšení označení na hodnotu 3, která zajistí vytvoření odpovídajícího výstupu. Pokud počet nepřesáhl hodnotu 2000, je ještě nutné zkontrolovat překročení hodnoty 150. To by znamenalo podezřelý tok a vedlo ke zvýšení označení na hodnotu 2. Obě tyto podmínky jsou rozšířeny o kontrolu aktuálního označení, aby nedocházelo ke snižování označení.

Pokud položka odpovídající záznamu neexistuje, je na konci pole vytvořena nová. Ta je typu `flow_record_t` a je zobrazena na obrázku 6.1. Musí také dojít k jejímu naplnění informacemi, takže se zapíše zdrojová a cílová IP adresa do `src_IP` a `dst_IP`, aktuální čas do `detect_time` i do `end_time`. Dále se uloží počty bajtů do `bytes` a paketů do `packets`, počet záznamů položky `flow_count` se nastaví na 1, stejně tak označení položky `level`. Počítadlo čistících cyklů `clean_count` se nastaví na 0.

V tomto případě, po přidání nové položky je nutné zkontrolovat, zda nebylo dosaženo hranice pole. Jeho aktuální velikost je uložena v proměnné `records_size` a pokud počet položek dosáhne této hodnoty je pole rozšířeno na dvojnásobnou velikost.

6.6 Zhodnocení implementace a návrhu

Celý vývoj proběhl do této fáze poměrně bez větších problémů. Za důvody této úspěšné a bezproblémové implementace považuji dobrý návrh. Druhým důvodem je potom možnost využití knihoven systému Nemea. V těch bylo možné nalézt několik maker, které usnadnily celý vývoj a dovolily mi soustředit se hlavně na problém detekce. Také jsou důvodem menšího rozsahu zdrojového kódu.

Kapitola 7

Testování

Vzhledem k účelu, za jakým bude modul používán, je důkladné testování velmi důležitou částí celé mé práce. I proto již během celého vývoje bylo provedeno několik testů, zda modul pracuje správně a nevyskytují se v něm chyby. Ovšem po dokončení finální verze je nutné cíleně a důkladně otestovat jak jednotlivé části modulu, tak i celek. Z tohoto důvodu jsem testování rozdělil na několik částí, které se soustředily na konkrétní problémy.

Protože se na modul vyskytují specifické požadavky, byla první část testování zaměřena na stabilitu modulu. Jejím cílem bylo ověřit, zda modul zvládne zatížení provozu v nekonečném čase. Zhodnocení těchto testů je popsáno v první části kapitoly. V další části je popsáno testování detekční části. Posledním problémem, který bylo nutné testovat, jsou falešné detekce.

7.1 Stabilita modulu

K testování tohoto problému bylo přistoupeno v první části, protože pokud by modul nebyl stabilní, nebylo by možné provádět další formy testování. Jelikož modul musí zvládnout pracovat v režimu označovaném jako 24/7, což znamená bez ustání, bylo nutné jej zatížit přibližně stejným objemem dat, jaký lze očekávat v reálném provozu.

Nejjednodušším způsobem bylo tedy modul připojit na kolektor, ze kterého přijímal data reálného provozu. Tyto data jsou získávána z desítky sond rozmístěných v síti CESNET, kde každá monitoruje 10Gb/s linku. Při tomto zatížení modul pracoval několik celodenních cyklů. Z těchto testů jsem získal dostatek dat, abych zjistil, jaké zatížení modul přibližně zpracovává. V tabulce 7.1 jsou uvedeny přibližné počty záznamů, procházejí přes modul v závislosti na době. Samozřejmě se tyto hodnoty mohou měnit, neboť zatížení se neustále mění podle vytížení sítě. Protože měření bylo prováděno několik dní, odpovídají reálným hodnotám v době vývoje modulu.

Po ukončení několikadenního testování stability modulu, jsem jej vyhodnotil jako stabilní, jelikož se neprojevily žádné chyby a objem dat přes něj proudící byl zpracováván bez problémů.

záznamů/24h	záznamů/1h	záznamů/1m
2,99 mld.	124,9 mil.	2,1 mil.

Tabulka 7.1: Výsledky testování stability modulu.

protokol	cílový port	délka záznamu	počet paketů	bajtů na paket	příznaky
61%	13%	13%	7%	4%	2%

Tabulka 7.2: Výsledky testování propustnosti filtru.

7.2 Testování filtrace

Podstatná část spolehlivé detekce je správně nastavený filtr, který bude propouštět správné množství dat, se kterými bude dále modul pracovat. Z tohoto důvodu jsem se rozhodl také zjistit účinnost navrženého filtru. Tento test jsem prováděl v době, kdy byl provoz na síti větší, abych získal dostatek dat. Počet záznamů, které při testu přes modul prošly, je přibližně 1,8 mld., což považuji za dostačující počet pro otestování propustnosti. Výsledky tohoto testu jsou zobrazeny v tabulce 7.2, kdy ke každé úrovni filtru je procentuálně uvedeno kolik záznamů se přes něj dostalo[1].

Z výsledků lze poznat, že propustnost filtru je malá, což by mělo být dobré pro detekci útoků, pokud nedochází i ke značné filtraci záznamů vznikajících při útoku. Tento problém se zřejmě nevyskytl podle výsledků dalších testů. Jediný problém, který se tímto testem projevil, je nízký podíl kontroly délky záznamu na celkovém množství vyfiltrovaných dat.

7.3 Spolehlivost detekce

Nejlepším způsobem, jak otestovat spolehlivost detekční části, bylo vystavení modulu reálným útokům. A protože z předchozích fází vývoje jsem měl všechny potřebné prostředky dostupné, zvolil jsem tuto variantu.

Při provádění útoku pomocí skriptu slowloris.pl, viz sekce 3.2, je možné zvolit velké množství nastavení. Tím vniká možnost pro velký počet útoků, které mají odlišné parametry. Z toho důvodu jsem vytvořil testovací skript, který spouští detektor a také provádí 13 útoků s různým nastavením. Nejčastěji různé kombinace počtu spojení a doby mezi oživováním těchto spojení. Dobu útoku jsem zvolil 30 min, což by měl být dostatek času k vytvoření množství záznamů, na jejichž základě modul bude útok detekovat.

Výsledky tohoto testování jsou uvedeny v tabulce 7.3. Ve sloupečcích timeout, interval oživování spojení v sekundách, a num, počet otevřených spojení je uvedeno nastavení útoku. Dále čas potřebný k detekci útoku. Poslední tři sloupečky jsou celkový počet vytvořených záznamů o útoku, počet záznamů, které modul vyfiltroval a uložil k danému toku a procento těchto uložených záznamů z celkového počtu vytvořených.

Před vyhodnocením výsledků je nutné uvést, že testování bylo několikrát opakováno, pro získání přesnějších výsledků a zjištění případných velkých rozdílů mezi nimi. Jelikož výsledky testů byly ve většině případů poměrně shodné. Lišily se počty záznamů, ovšem procento zachycených se měnilo jen málo. Dále se lehce měnila doba potřebná k detekci, což souvisí s tím jak rychle modul obdrží záznamy. Žádná z těchto změn ovšem nebyla nijak výrazná, aby bylo nutné se nad ní pozastavit a hledat problém. Nejvýraznější změna potom byla, zda modul detekoval útok při testu č.8, jelikož hodnoty se zde pohybovaly vždy v okolí detekční hranice modulu. Proč tomu tak je, je vysvětleno níže.

Z celkového počtu 13 testů tedy modul detekoval 11 útoků, což je přibližně 85% úspěšnost. Doba potřebná k detekci u testu č.3 je dlouhá, ale to je dáno nastavením útoku, který

[1] Procenta jsou vypočtena z celkového počtu záznamů.

Číslo útoku	timeout [s]	num	Doba detekce	Celkový počet záznamů	Zachycený počet záznamů	% zachycených záznamů
1	10	300	-	1434	0	0%
2	30	300	0:07:07	10212	10101	99%
3	200	300	0:25:09	2276	2032	89%
4	5	500	-	1896	0	0%
5	60	500	0:13:50	7214	6775	94%
6	120	500	0:08:38	6938	6659	96%
7	150	500	0:07:14	7489	7111	95%
8	10	700	0:07:47	4451	2060	46%
9	30	700	0:07:08	9352	8701	93%
10	60	1000	0:08:20	8433	7083	84%
11	100	1000	0:06:32	8308	7973	96%
12	200	1000	0:13:18	4532	4397	97%
13	30	1200	0:07:08	10678	8439	79%

Tabulka 7.3: Výsledky testování spolehlivosti detektoru.

vytváří málo toků a tím i záznamů potřebných k detekci. To lze vidět v dalších sloupečcích, kde celkový počet toků jen těsně přesáhl detekční hranici 2000 během 30 minut testování.

Při neúspěšných testech č.1 a č.4, vzhledem k nastavení útoků, vznikají záznamy z velké části naprosto odlišné, či přímo opačné než jaké přicházejí v jiných případech. K tomu dochází, protože intervaly mezi aktivitou na spojení jsou nastaveny na kratší dobu než 30s, což je hodnota nastavení neaktivního timeoutu na sondách. Z toho důvodu dochází k agregaci toků a tím vznikají odlišné záznamy. Proto je zde procento uložených záznamů z celkového počtu vytvořených nulové. Avšak při nastavení většího počtu spojení, viz test č.8, se zvyšuje intenzita útoku a tím vzniká větší počet záznamů. I zde je část záznamů z důvodu agregace velmi odlišná, což dokazuje nižší procento zachycených záznamů.

V ostatních případech testů výsledky značí poměrně úspěšnou detekci, která často netrvá déle jak 10 minut. Doba detekce je často závislá na intenzitě útoku, proto přesnějším ukazatelem je procento uložených z celkového počtu vytvořených záznamů o útoku, které se u většiny testů pohybuje nad hranicí 90%.

7.4 Falešné detekce

Testování modulu na falešné detekce nakonec nebylo třeba vytvářet, jelikož předchozí formy testování víceméně kontrolovaly jejich výskyt. K vyhodnocení této části testování jsem tedy použil informace získané z předchozích testů. V těch jsem nenašel žádný záznam o detekci útoku. Protože se jednalo o přibližně 140 hodin testování, zvolil jsem tuto formu jako dostačující a k falešným detekcím by tedy nemělo docházet.

Kapitola 8

Závěr

V úvodu své práce jsem se zabýval teoretickou částí, kdy jsem studoval problematiku počítačových útoků typu DoS a dále jsem pokračoval zaměřením se na variantu Slowloris, která je základem mé práce. Prozkoumal jsem charakteristiku těchto útoků, nástroje dostupné pro jeho provádění a získal další informace potřebné k detekci. Dále jsem se více zaměřil na jeden z nejvíce používaných nástrojů k provedení tohoto typu útoku a to skriptu slowloris.pl. Protože výsledkem byl detektor toho útoku ve formě modulu pro systém Nemea, musel jsem se s tímto systémem v závěru teoretické části seznámit.

V praktické části práce jsem analyzoval záznamy, které jsou vytvářeny kolektory při útoku pomocí skriptu slowloris.pl. Na základě výsledků této analýzy a poznatků o systému Nemea jsem mohl vytvořit návrh modulu a přistoupit k jeho implementaci. V závěru praktické části bylo nedílnou součástí práce otestování vytvořeného modulu. Tyto testy ukázaly výbornou stabilitu modulu, ale také poměrně dobrou spolehlivost detekční části proti Slowloris útokům.

Protože všechny testy detekční části nebyly úspěšné, vzniká zde prostor pro možnost dalšího rozšíření práce. Při dalším pokračování je možné se zaměřit na detekci krajních variant Slowloris útoků, jež se vyznačují odlišně než větší část pokrytá touto prací. Vzhledem k počtu toků, které tyto spojení vytvářejí je jejich detekce velmi složitá, neboť tyto toky často vypadají jako toky normálního provozu. Také by zřejmě byl potřebný dlouhý čas k detekci. Takového rozšíření by se dalo využít spíše ke zpětné kontrole zaznamenaných dat na přítomnost útoku. Dále také rozšíření k pokrytí dalších modifikací Slow HTTP DoS, například tzv. pomalé čtení, Slow Post DoS. Tyto varianty nebyly při vývoji modulu pokryty, protože bylo zjištěno, že k jejich detekci je potřebné zvolit jinou formu zpracování.

Během celého vývoje jsem získal mnoho znalostí z oblasti síťových útoků, detailnější pak o typech DoS. Seznámil jsem se s projektem Nemea a využitím záznamů o tocích při analýze a detekci v síťovém provozu. Celá práce pro mě byla velmi prospěšná a získal jsem při ní mnoho nových zkušeností.

Literatura

[1] Using Denial of Service for Hacking [online]. http://ha.ckers.org/blog/20090504/using-denial-of-service-for-hacking/, 2009-05-07 [cit. 2014-05-11].

[2] Slowloris HTTP DoS [online]. http://ha.ckers.org/slowloris/, 2009-06-17 [cit. 2014-05-11].

[3] Bartoš, V.; Žádník, M.; Čejka, T.: Nemea: Framework for stream-wise analysis of network traffic. http://www.cesnet.cz/wp-content/uploads/2014/02/trapnemea.pdf, 2013-12-13 [cit. 2014-05-11].

[4] Dittrich, D.; Mirkovic, J.; Reiher, P.; aj.: *Internet Denial of Service: Attack and Defense Mechanisms*. Pearson Education, 2004, iSBN 0-13-147573-8.

[5] Haller, M.: Denial of Service (DoS) útoky: úvod [online]. http://www.lupa.cz/clanky/denial-of-service-dos-utoky-uvod/, 2006-09-05 [cit. 2014-05-11].

[6] Hlaváček, T.: DoS a DDoS útoky na vzestupu cílených útoků [online]. http://diit.cz/blog/dos-a-ddos-utoky-na-vzestupu-cilenych-utoku, 2013-12-12 [cit. 2014-05-11].

[7] Krčmář, P.: Útok Slowloris aneb plíživé nebezpečí pro web servery [online]. http://www.root.cz/clanky/utok-slowloris-aneb-plizive-nebezpeci-pro-web-servery/, 2011-05-17 [cit. 2014-05-11].

[8] Macháček, M.: Počítačová kriminalita a bezpečnost [online]. http://www.internetprovsechny.cz/pocitacova-kriminalita-a-bezpecnost/, 2013-04-02 [cit. 2014-05-11].

[9] Malcolm, N.: mod_antiloris - Anti Slowloris Apache Module [online]. https://coderwall.com/p/hmgy3q.

[10] Pillai, S.: SLOWLORIS: HTTP DOS(Denial Of Service)attack and prevention [online]. http://www.slashroot.in/slowloris-http-dosdenial-serviceattack-and-prevention, 2013-02-24 [cit. 2014-05-11].

[11] Prosise, C.; Mandia, K.: *Počítačový útok Detekce, obrana a okamžitá náprava.* COMPUTER PRESS, 2002, iSBN 80-7226682-9.

[12] Senecal, D.: Slow DoS on the Rise [online].
`https://blogs.akamai.com/2013/09/slow-dos-on-the-rise.html`, 2013-09-12 [cit. 2014-05-11].

[13] Shekyan, S.: Testing Web Servers for Slow HTTP Attacks [online].
`https://community.qualys.com/blogs/securitylabs/2011/09/19/testing-web-servers-for-slow-http-attacks`, 2011-09-19 [cit. 2014-05-11].

[14] Shekyan, S.: How to Protect Against Slow HTTP Attacks [online].
`https://community.qualys.com/blogs/securitylabs/2011/11/02/how-to-protect-against-slow-http-attacks`, 2011-11-02 [cit. 2014-05-11].

[15] Shekyan, S.: Are you ready for slow reading? [online].
`https://community.qualys.com/blogs/securitylabs/2012/01/05/slow-read`, 2012-01-05 [cit. 2014-05-11].

Příloha A

Obsah CD

- Technická zpráva ve formátu PDF.
- Zdrojové LaTeXsoubory technické zprávy.
- Instalační balík systému Nemea, včetně modulu.
- Návod k instalaci README.txt.
- Zdrojový soubor modulu.

Příloha B

Manual

Po stažení instalačního balíčku z CD je nutné jej rozbalit, příkazem:

```
tar xzf nemea-1.0.0.tar.gz
```

Dále se přesunout do adresáře systému a spustit configuraci, pomocí příkázů:

```
cd nemea-1.0.0
./configure --prefix=/usr
```

Poté je nutné provést překlad a instalaci sytému, příkazy:

```
make -C libtrap
sudo make -C libtrap install
make -C unirec
```

Tím je dokončena instalace a všechny operace potřebné ke zprovoznění modulu. Dále tedy vstoupení do složky modulu a překlad:

```
cd modules/slow_http_detector
make
```

Při spouštění modulu je nutné parametrem zadat informace kam se mají připojit rozhraní. K tomu slouží parametr `-i "tt;adresa_vstup,port_vstup;port_výstup"`. Např.:

```
./slow_http_detector -i "tt;localhost,12345;54321"
```

Tím je nastaveno, že vstup modulu se připojí na `localhost:12345` a výstup na `54321`.