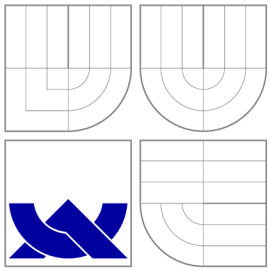
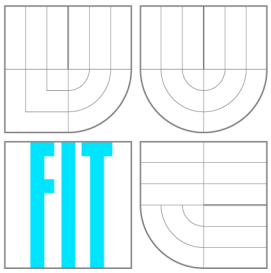


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DETEKCE ÚTOKŮ TYPU SYN FLOOD

DETECTION OF SYN FLOOD ATTACKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL RUPRICH

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VÁCLAV BARTOŠ

BRNO 2014

Abstrakt

Tato práce se zabývá detekcí anomálií v analýze síťového provozu. Cílem je implementace tří algoritmů, určených pro odhalení síťových útoků typu SYN flood. Použité metody monitorují síťový provoz v reálném čase a vytváří určitý model běžného chování provozu. Tento model pak slouží k odhalení chování, které do modelu nezapadá a je tak považováno za anomálii. Algoritmy byly implementovány v programovacích jazycích C a C++. Nastavení parametrů algoritmů a jejich testování bylo založeno na reálných datech z monitorovacích sond organizace CESNET s použitím frameworku Nemea.

Abstract

The thesis deals with a topic of anomally detection in network traffic. The goal is to implement three algorithms which will be able to reveal SYN flooding types of network attacks. Used methods monitor network traffic in real time and create certain model of normal traffic behaviour. This model is then used to detect behaviour which does not fit the model and therefore is considered as an anomally. Algorithms were implemented in C and C++ programming languages. Settings of parameters and testing of algorithms is based on real data gathered from monitoring probes deployed by CESNET organization by using the Nemea framework.

Klíčová slova

Detekce anomálií, DoS útok, SYN flooding

Keywords

Anomaly detection, DoS attack, SYN flooding

Citace

Michal Ruprich: Detekce útoků typu SYN flood, bakalářská práce, Brno, FIT VUT v Brně, 2014

Detekce útoků typu SYN flood

Prohlášení

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

.....
Michal Ruprich
19. května 2014

Poděkování

Na tomto místě bych velice rád poděkoval vedoucímu této práce Ing. Václavu Bartošovi především za podnětné rady a připomínky.

Obsah

1 Úvod **4**

2 Bezpečná síť **5**
2.1 Klasický přístup . . . 5
2.2 Systém pro detekci narušení sítě . . . 5
2.2.1 Detekce signatur . . . 6
2.2.2 Detekce anomálií . . . 6
2.2.3 Hybridní systémy . . . 7
2.3 Detekce anomálií v síťovém provozu . . . 7
2.3.1 Detekce anomálií za pomoci statistiky . . . 7

3 TCP SYN flooding **9**
3.1 Protokol TCP . . . 9
3.1.1 Zahájení TCP komunikace . . . 9
3.1.2 Ukončení TCP komunikace . . . 9
3.2 Principy SYN flooding . . . 10
3.2.1 DoS a DDoS . . . 10
3.2.2 SYN flooding . . . 11
3.3 Techniky SYN flooding útoku . . . 11
3.3.1 Přímý útok . . . 11
3.3.2 Spoofing . . . 12
3.3.3 Distribuovaný útok . . . 12
3.4 Možná protiopatření . . . 13

4 Detekce SYN flooding útoků **15**
4.1 Metoda SynFinDiff . . . 15
4.1.1 Dvojice SYN-FIN (RST) . . . 15
4.1.2 CUSUM algoritmus . . . 16
4.2 Metoda Synrate . . . 17
4.2.1 Adaptive threshold algoritmus . . . 17
4.2.2 CUSUM algoritmus . . . 17
4.3 Metoda Partial Completion Filter . . . 18

5 Implementace **20**
5.1 Nemea framework . . . 20
5.2 Moduly . . . 21
5.2.1 Synfindiff . . . 22
5.2.2 SynRate . . . 22

5.2.3 PCF 23

6 Testování **25**
6.1 Synfindiff 25
6.2 Synrate 28
6.3 PCF 30
6.4 Shrnutí 31

7 Závěr **32**

A Obsah CD **34**

Seznam obrázků

3.1 TCP three-way a four-way handshake 10
3.2 SYN spoofing 12
3.3 Distribuovaný SYN flooding 13

4.1 Partial Completion Filter 19

6.1 Zapojení modulů pro testování v rámci systému Nemea 25
6.2 Porovnání nastavení parametru α u metody Synfindiff 26
6.3 Délka monitorovacího okna nastavena na 40s 27
6.4 Délka monitorovacího okna nastavena na 160s 27
6.5 Ukázka monitorování algoritmem Synfindiff při útoku SYN flood 28
6.6 Hodnoty SYN a EWMA algoritmu Synrate s různým nastavením parametru β 29
6.7 Synrate při nastavení $\beta = 0,2$ 30
6.8 Synrate při nastavení $\beta = 0,9$ 30

Kapitola 1

Úvod

Globální internetová síť dnes propojuje téměř všechny kouty světa a už dávno se stala neodmyslitelnou součástí každodenního života. Data, informace a služby jsou přístupné 24 hodin denně pomocí pouhých několika kliknutí. Mnoho firem dnes disponuje vlastní firemní sítí, pomocí které poskytují služby svým zákazníkům a usnadňují práci svým zaměstnancům. Přístup k těmto sítím je často možný jednoduše přes internet, ať už se jedná o zaměstnance, který vzdáleně přistupuje k firemním datům, nebo o zákazníka, který využívá služeb firmy dostupných online. Síť se však stává zranitelná vůči napadení.

Napadení sítě dnes dávno není doménou pouze počítačových specialistů a hackerů. Programy, které jsou schopné vytvořit síťový útok, se dají nalézt online ke stažení a použít je může téměř kdokoli. O to víc je potřeba, aby každá síť byla zabezpečena proti různým druhům napadení. Problémem je, že i přes všechna bezpečnostní opatření není vždy možné síť zabezpečit na 100 procent.

Algoritmy, vytvářené za účelem detekce anomálií v síťovém provozu, by se daly přirovnat k poplašným systémům v reálném životě. Z monitorování a analýzy síťového provozu je možné vysledovat rysy a charakteristiky, kterými se provoz v danou denní dobu vyznačuje. Pomocí těchto informací je možné vytvořit jakýsi model chování síťového provozu. Detekční algoritmus pak využívá tento model pro určení, zda se v síťovém provozu nevyskytují anomálie, které indikují narušení sítě.

Cílem práce je implementace tří algoritmů pro detekci anomálií v síťovém provozu a jejich otestování a porovnání na základě reálných síťových dat. Pokud žádná z metod neumožňuje detekovat zdroj nebo cíl útoku, měl by být navržen způsob, jak to umožnit.

Práce je v zásadě rozdělena na dvě části. První část se zabývá především teoretickým základem nutným pro pochopení problematiky detekce anomálií a SYN floodingu. Nejdříve je uvedeno vysvětlení, co jsou vlastně systémy pro detekci narušení sítě, k čemu jsou takové systémy dobré a jaké různé druhy systémů se dnes využívají. V druhé kapitole je pak vysvětlen princip fungování protokolu TCP a jak je tento princip zneužit právě při napadení sítě útokem typu SYN flood. V další kapitole jsou podrobně popsány tři algoritmy, které byly vytvořeny za účelem detekce právě SYN flood útoků.

Druhá část práce je zaměřena na praktickou stránku věci. Jsou zde dvě kapitoly zaměřené na implementaci algoritmů a na jejich otestování na reálných síťových datech. V implementační části je popsán systém, na kterém byly algoritmy vyvíjeny a jsou zde uvedeny implementační detaily samotných algoritmů tak, jak byly vyvíjeny v programovacích jazycích C a C++. Kapitola o testování má ukázat například, jakým způsobem se vytváří model chování síťového provozu a jakou roli zde hrají některé základní parametry algoritmů.

Kapitola 2

Bezpečná síť

2.1 Klasický přístup

Mezi klasické přístupy využívané pro zabezpečení sítě před neoprávněným zásahem by se daly zařadit antiviry jako ochrana koncových zařízení, ať už se jedná o osobní počítače nebo servery, a firewally jako ochrana tzv. *perimetru* sítě. Perimetr sítě slouží jako jakási pomyslná hranice oddělující vnitřní, relativně bezpečné prostředí sítě LAN, a vnější, potenciálně nebezpečné prostředí WAN.

Antivirové programy obsahují databázi známých škodlivých programů, umí je vyhledat v souborovém systému a následně zneškodnit. Tato ochrana se však týká výhradně koncových zařízení a není uzpůsobena pro širší ochranu sítě nebo podsítě. Navíc antivirová databáze obsahuje pouze již známé typy útoků a jakýkoli nový způsob zásahu neumí odhalit.

Firewall je softwarový nebo hardwarový systém, který sleduje a analyzuje síťový provoz a na základě předem stanovených pravidel rozhoduje, zda je nutné konkrétní komunikaci zablokovat nebo ne. Nedokonalostí firewallu se však může stát jakákoli chyba nebo špatný předpoklad, které vzniknou při návrhu, implementaci nebo konfiguraci. Firewall lze také obejít tak, že útočník využije běžně používaných internetových protokolu jako HTTP nebo FTP a útok tak zamaskuje za běžnou síťovou komunikaci. Za předpokladu, že se někomu podaří odhalit slabinu a dostat se tak za perimetr sitě, firewall jakoby v tu chvíli ani neexistoval a útočník má volnou ruku.

V neposlední řadě představují bezpečnostní riziko i samotní uživatelé sítě, kteří nejsou seznámeni s případnými riziky a jsou schopni nevědomky umožnit útočníkovi neautorizovaný přístup nebo zadní vrátka od sítě.

2.2 Systém pro detekci narušení sítě

Systémy pro detekci narušení sítě neboli *Intrusion Detection Systems – IDS* bychom mohli přirovnat k poplašným zařízením v reálném světě. Nastane-li situace, kdy dojde k narušení perimetru sítě, systém by měl narušení odhalit a upozornit na něj, případně odhalit zdroj napadení. Pokud bychom měli srovnat funkci firewallu a IDS, můžeme říci, že firewall se snaží zabránit, aby k útoku vůbec došlo, zatímco IDS hraje roli ve chvíli, kdy už útok probíhá. IDS sleduje nejenom síťovou komunikaci proudící směrem dovnitř sítě, ale i komunikaci probíhající uvnitř nebo proudící směrem ven ze sítě. Je tedy zřejmé, že IDS není systém, který by snad měl nahradit stávající ochranné prvky sítě, ale spíše je doplnit a společně tak vytvořit robustní zabezpečení.

V [7] rozdělují IDS systémy na tři základní skupiny podle toho, jakým způsobem klasifikují síťový provoz a podle čeho se pak rozhodují, zda došlo k narušení:

- Detekce signatur
- Detekce anomálií
- Hybridní systémy

Každý z těchto přístupů má své výhody i nevýhody, popsané v následujících odstavcích.

2.2.1 Detekce signatur

Systém pro detekci signatur monitoruje jednotlivé pakety v síti a porovnává jejich obsah s databází, která obsahuje typické rysy již známých technik používaných k narušení sítě. Tento systém funguje podobným způsobem, jakým antivirový software odhaluje malware na koncových zařízeních. Parametry, které hrají roli při identifikaci neoprávněných zásahů, mohou být například kombinace polí z IP a TCP hlavičky paketu (typ protokolu, cílový port) společně s daty z datové části paketu. Pokud se ukáže, že paket vykazuje podobné rysy s některým záznamem v databázi, systém ohlásí pokus o útok a paket dál nepustí.

Velikou výhodou tohoto přístupu je nízká míra falešných poplachů (false positives). Je to především proto, že systém díky porovnání paketů s databází okamžitě „ví“, co klasifikovat jako podezřelé a co jako normální.

Nevýhodou však je, že aby mohl systém úspěšně chránit před všemi možnými útoky, musí jeho databáze obsahovat všechny možné rysy, kterými se mohou vyznačovat. V případě, že se objeví nový druh útoku, jehož signatura zatím není známa, systém na ni neumí zareagovat a není tak schopen účinně chránit síť. Tyto útoky nazýváme také *zero-day* útoky. Okno zranitelnosti pak označuje dobu, která proběhne mezi prvním použitím útoku a aktualizací databáze tak, aby byl systém schopen útok odhalit. Další nevýhodou se pak může stát fakt, že systém musí prozkoumat každý průchozí paket, čímž se může zpomalit průtok dat.

2.2.2 Detekce anomálií

Systém pro detekci anomálií se spíše než na jednotlivé pakety zaměřuje na série paketů, ze kterých se snaží vyčíst jakési obecné chování síťového provozu. Hovoříme o tzv. behaviorálním přístupu monitorování sítě. Systém sleduje chování sítě za normálních podmínek, ze kterých se „naučí“, jak vypadá normální chování. Jakékoli odchylky od normálního chování jsou pak považovány za podezřelé a systém je může vyhodnotit jako případný útok.

Systém má oproti předešlému přístupu několik výhod. V prvé řadě se výrazně zmenšuje problém zero-day útoků, protože cokoliv, co svým chováním vybočuje, je bráno jako potenciální hrozba. Další výhodou je jistá soběstačnost systému v tom smyslu, že není potřeba udržovat a obnovovat databázi možných ohrožení jako v předešlém případě. Navíc systém je možné implementovat téměř kdekoli, protože se sám přizpůsobí pro potřeby konkrétní sítě.

Problémem při implementaci takovéhoto způsobu detekce však nastává ve chvíli, kdy je potřeba určit, co je „normální“ chování. S tím souvisí i vyšší míra falešných poplachů než v předchozím případě, protože i výrazné abnormality v chování nemusí nutně znamenat narušení sítě. A naopak nepřítomnost výchylek běžného provozu nemusí být známkou, že je vše v pořádku vzhledem k existenci útoků, jejichž účinek se nemusí nutně projevit. Stejně

jako v předchozím případě se monitorování tohoto typu může značně projevit na výkonu sítě. Právě algoritmy, které mají za úkol detekovat anomálie v síti jsou hlavní náplní této práce.

2.2.3 Hybridní systémy

Hybridní systém je takový, který má snahu o kombinování pozitiv a zároveň eliminaci negativ předešlých systémů. Teoreticky by takový systém poskytoval mnohem robustnější obranu, vytvořit jej však je velmi složité a ve výsledku ani nemusí být nutně lepší. Mezi nejtěžší úkoly takového přístupu patří především funkční a efektivní propojení jednotlivých technologií a postupů. Typicky se zde kombinuje detekce anomálií pro neznámé techniky a detekce signatur, která napomáhá v odhalení již známých útoků.

2.3 Detekce anomálií v síťovém provozu

Podle [12] lze systém pro detekci anomálií formálně definovat jako $S = (M, D)$, kde M definuje model normálního chování a D označuje míru odchýlení od modelu M vzhledem ke sledované aktivitě. M lze tedy označit jako modelovaný subsystém a D jako odchylkový subsystém (deflection subsystem).

Oba subsystémy fungují částečně odděleně. M je vytvářen především během učebního procesu algoritmu, kdy se vytváří model normálního chování pro konkrétní sledovanou aktivitu. Měl by být periodicky obnovován, aby se model přizpůsobil případným změnám a výkyvům v síťovém provozu. D pak definuje, jaké chování je možno považovat za odchýlení od M až do té míry, že se jedná o útok.

[8] udává několik základních přístupů, které definují způsob sledování konkrétních síťových aktivit:

- Detekce anomálií za pomoci statistiky
- Detekce anomálií pomocí strojového učení
- Detekce anomálií na základě techniky data mining

Algoritmy implementovány v rámci této práce patří do skupiny detekce anomálií za pomoci statistiky. Ostatní skupiny přesahují rámec této práce.

2.3.1 Detekce anomálií za pomoci statistiky

Systém sleduje konkrétní síťové aktivity a vytváří dva základní profily, které reprezentují jejich chování. Jsou to aktuální profil, vypočítaný z hodnot sledované aktivity naměřených v aktuálním sledovacím období, a předešlý profil, naměřený v předchozím sledovacím období. Systém periodicky přepočítává aktuální profil a vypočítává tzv. hodnotu odchýlení (anomaly score) tak, že porovná aktuální profil s předchozím profilem za použití funkce abnormality všech předchozích měření. Varování, že došlo k útoku, se pak řídí podle předem určeného mezníku (threshold). Pokud je threshold hodnotou odchýlení překročen, znamená to výskyt anomálie.

Výhodou takového přístupu je především fakt, že systém nemusí přesně znát povahu konkrétních útoků, jejich přítomnost vysleduje z chování konkrétní aktivity. Navíc je takový systém schopen detekovat útoky, které jsou typické tím, že probíhají delší časový úsek.

Typickým příkladem takového útoku je *portscan*, který se obvykle pozná podle výrazného zvýšení síťového provozu v porovnání s běžnou komunikací.

Nevýhodou je, že detekční systém, založený na statistice, je teoreticky možné oklamat, pokud má útočník dostatek informací o tom, jaké obranné mechanismy jsou v síti implementovány. Detekční algoritmus by se měl být schopen adaptovat na případný nárůst legitimního provozu v síti a neklasifikovat tento nárůst jako hrozbu. Pokud tedy útočník svůj útok stupňuje během delšího časového období, detekční systém si může na tento nelegitimní provoz „zvyknout“ a nehlásit jej jako anomálii. Navíc je zde problém, že některé nežádoucí chování jednoduše nelze popsat pouze pomocí statistiky a je třeba přistoupit k jiným metodám.

Kapitola 3

TCP SYN flooding

Tato kapitola vychází především z normy RFC č. 793 ([9]), která popisuje základní parametry spojované TCP komunikace a z normy RFC č. 4987 ([4]), která se zabývá základními charakteristikami problematiky SYN flooding útoků.

3.1 Protokol TCP

TCP je síťový protokol zajišťující tzv. spojovanou síťovou komunikaci. Spojovaná komunikace zajišťuje bezztrátovost odeslaných dat pomocí zasílání potvrzení o přijetí a zároveň zaručuje, že data budou po přijetí seřazena ve správném pořadí. Aby mohla spojovaná komunikace správně probíhat, je třeba nejdříve mezi serverem a klientem navázat spojení předtím, než se začnou posílat data. Technika kterou k tomu využívá protokol TCP se nazývá *three-way handshake*.

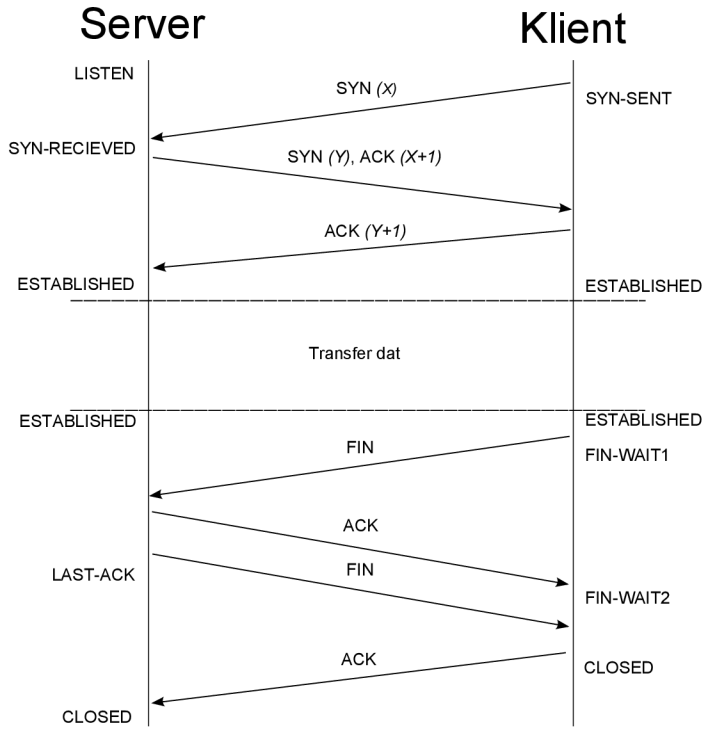
3.1.1 Zahájení TCP komunikace

Na obrázku 3.1 je vidět proces ustálení komunikačního kanálu mezi klientem a serverem. Klient zahájí komunikaci odesláním tzv. synchronizačního paketu, který má v hlavičče TCP nastaven příznak SYN a proces se přesune do stavu SYN-SENT. Tímto říká serveru, že chce navázat spojení a začít komunikaci. Server na přijatý SYN odpoví paketem s nastavenými příznaky SYN/ACK a přejde do stavu SYN-RECEIVED. Klient po příjetí SYN/ACK odešle paket s nastaveným příznakem ACK. Když jej server přijme, server i klient přechází do stavu ESTABLISHED a spojení je navázáno.

Každé TCP spojení si uchovává určité informace jako jsou stav spojení, lokální proces obsluhující spojení a další potřebné parametry. Během procesu ustálení komunikace, typicky ve stavu SYN-RECEIVED, se pro každé nové připojení alokuje blok paměti (TCB – Transmission Control Block), ve kterém jsou tyto informace uloženy. Samostatný TCB je vytvořen pro každou novou TCP relaci.

3.1.2 Ukončení TCP komunikace

Proces ukončení spojení TCP se nazývá *four-way handshake* a jak je možno vidět na obrázku 3.1, probíhá ve čtyřech krocích. Jedna z komunikujících stran, která chce spojení ukončit, odešle paket s nastaveným příznakem FIN a přechází do stavu FIN-SENT1. Druhá strana potvrdí pomocí ACK, pošle FIN paket a přejde do stavu LAST-ACK, ve kterém nějaký čas vyčkává na přijetí posledního potvrzení ukončení relace. První strana po obdržení ACK



Obrázek 3.1: TCP three-way a four-way handshake

přechází do stavu FIN-WAIT2, dokud neobrží FIN paket. Nakonec ukončení potvrdí posledním ACK a obě strany mohou spojení uzavřít. TCB obsahující informace o konkrétním spojení je uvolněn.

TCP spojení může být navíc ukončeno, pokud jedna strana odešle paket s nastaveným příznakem RST. Důvodem může být například chyba na straně, která RST paket odesílá. Spojení by mělo být ukončeno, pokud k tomu již nedošlo.

3.2 Principy SYN flooding

3.2.1 DoS a DDoS

SYN flooding neboli záplava SYN paketů se řadí mezi tzv. DoS (Denial of Service) nebo DDoS (Distributed Denial of Service) útoky.

DoS útoky se zaměřují na zahlcení linky nebo služby nevyžádanou komunikací, která má za následek její výrazné zpomalení, nebo i dočasnou nedostupnost pro legitimní uživatele. Takto napadený server nemůže nadále obsluhovat nová připojení a dochází ze strany serveru k tzv. odmítnutí služby (denial).

DDoS útoky jsou DoS útoky prováděné v mnohem větším měřítku, jsou tzv. distribuované, tedy zdroj útoku není pouze jeden. Útočník do útoku zapojí více koncových stanic, které na cílový server posílají obrovské množství dat. Toho dosáhne například tak, že vy-

tvoří program zvaný bot, který se může do nezabezpečeného počítače dostat jako malware. Zde vyčkává dokud není aktivován. Po aktivaci provádí zadané úkoly jako například zasílaní dat po internetu. Pokud se útočníkovi podaří rozšířit program na větší množství počítačů, může je využít pravě pro DDoS útok. Často se může jednat o stovky až tisíce počítačů, jejichž majitelé si takového zneužití vůbec nemusí být vědomi. Útok pak přichází z obrovského množství IP adres, zátěž pro napadený systém se tak stává neúnosná a jen velmi obtížně se odhaluje zdroj napadení.

3.2.2 SYN flooding

Jak bylo řečeno v 3.1, pro každé nové spojení je třeba na straně serveru vytvořit nový TCB. Mohou existovat rozdíly v tom, jakým způsobem jsou TCB implementovány a vytvářeny v různých operačních systémech, ale základem je, že každý TCP proces vyžaduje část paměti pro uchování stavových informací o vytvářené relaci. V praxi většina operačních systémů umí obsluhovat více připojení na jeden port zároveň. Za každé připojení to tak znamená alokování nové části paměti.

Aby tak nedošlo k vyčerpání paměti příliš velkým množstvím alokovaných TCB, udržují si operační systémy tzv. *backlog* záznam. V něm jsou zaznamenány všechny procesy, které jsou tzv. *half-open*, tedy ve stavu SYN-RECEIVED, který označuje nedokončenou synchronizaci. Backlog má nastavený maximální počet procesů ve stavu SYN-RECEIVED, které mohou probíhat zároveň. Pokud je dosáhnuto maxima, nejsou přijímány žádné nové připojení, dokud se neukončí některé z již inicializovaných. Zde však vyvstává nový problém a tím je vyčerpání kapacity backlog.

Technika SYN flooding využívá právě tento nedostatek v synchronizačním procesu TCP spojení. Útok je veden tak, že se na server odešle velké množství synchronizačních paketů, ale proces synchronizace je ze strany útočníka nedokončen. Server pro každý synchronizační paket, který obdrží, vytvoří nový TCB, který čeká na přijetí ACK a dokončení synchronizačního procesu. Nový proces navíc nějakou dobu vyčkává na přijetí ACK, než se ukončí (timeout) a uvolní výpočetní zdroje. Potvrzení synchronizace však ze strany útočníka neproběhne a vzniká tak veliké množství procesů, které jsou ve stavu SYN-RECEIVED a velmi brzy se zaplní backlog záznam. Každý nový požadavek na server, ač legitimní, je v takovém případě odmítnut.

3.3 Techniky SYN flooding útoku

Existuje několik způsobů popsaných v [5], jak provést SYN flooding útok. V zásadě je možné je rozdělit na tyto tři:

- Přímý útok
- Spoofing
- Distribuovaný útok

3.3.1 Přímý útok

Přímý útok může být proveden z jednoho koncového zařízení. Útočník se snaží otevřít veliké množství relací se serverem zároveň, čehož může dosáhnout například pomocí volání *connect()*. Je potřeba mít správně nastavený systém, aby neodpovídal na výzvy SYN/ACK

od serveru. Toho lze dosáhnout například nastavením filtrování na routeru. Obrana před tímto typem útoku je poměrně snadná. Pokud dojde k odhalení SYN flood útoku, útok přichází pouze z jedné IP adresy, kterou je třeba blokovat pomocí firewallu.

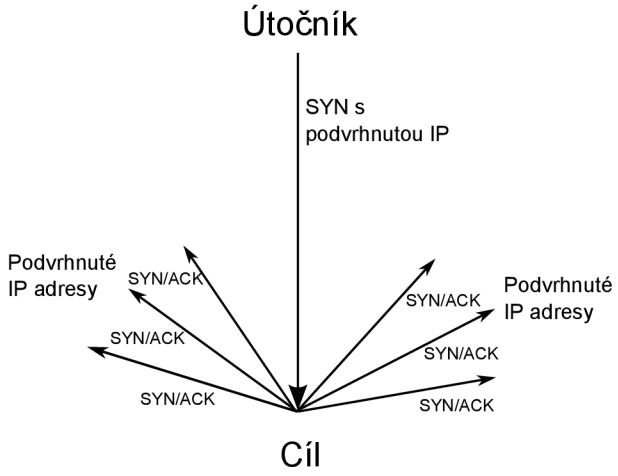
3.3.2 Spoofing

Spoofing obecně označuje metodu podvržení informací v síťovém provozu. V tomto případě se jedná o tzv. IP spoofing, neboli změnu zdrojové IP adresy v IP hlavičce. Odpověď na takový přijatý paket bude tedy odeslána právě na podvrženou IP adresu (viz. obrázek 3.2).

Klíčovým aspektem je volba IP adres, které budou použity místo pravé. V zásadě existují dvě možnosti:

1. Útočník zvolí jednu konkrétní adresu o které ví, že na výzvu SYN/ACK nebude odpovídat anebo rovnou neexistuje. Tuto adresu pak použije jako zdrojovou místo vlastní adresy. V takovém případě je ochranou stejný postup jako v 3.3.1.

2. Útočník zvolí více zdrojových adres s předpokladem, že velké procento z nich nebude na SYN/ACK reagovat. Zamezit útoku v tomto případě je složitější, protože není jasné, odkud útok konkrétně přichází.



Obrázek 3.2: SYN spoofing

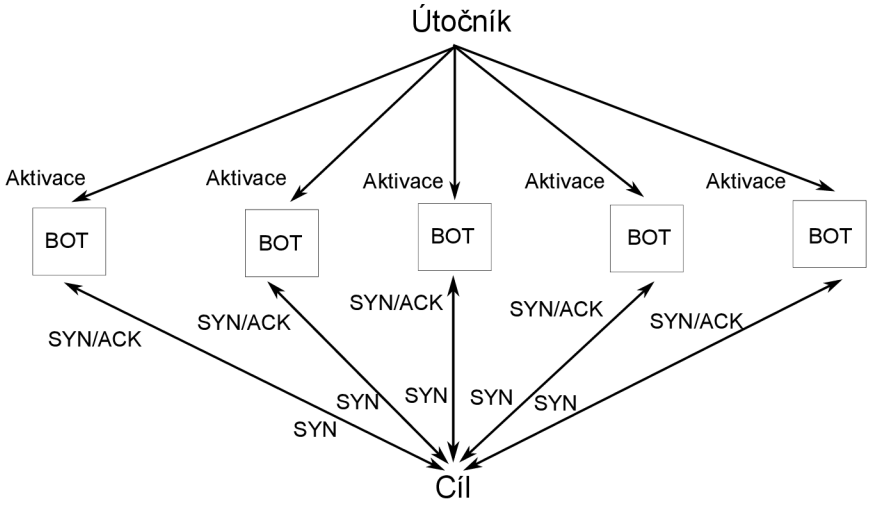
3.3.3 Distribuovaný útok

Distribuovaný SYN flooding probíhá podobně jak je popsáno v 3.2.1. Opět se nabízí dva způsoby provedení:

1. Každý z počítačů ovládaných botem posílá SYN pakety se svou zdrojovou IP adresou.

2. Každý z počítačů ovládaných botem posílá SYN pakety s podvrženou adresou. Vzniká tak vícenásobný spoofing.

Zabránit distribuovaným SYN flooding útokům je velmi obtížné.



Obrázek 3.3: Distribuovaný SYN flooding

3.4 Možná protiopatření

Existuje několik základních technik navržených v [4], které je možné využít jako obranné mechanismy proti SYN flooding útokům.

Větší backlog – Jako první možnost se může jevit zvětšení velikosti backlog záznamu, aby byl schopen pojmout co nejvíce možných připojení. Toto řešení je však nedostačující, protože i pokud je systém schopen efektivně udržovat veliký backlog, například distribuovaný útok jej snadno vyčerpá.

Recyklace backlog – Po naplnění backlog se nejstarší záznamy half-open TCB přepisují novými. Předpokládá se, že legitimní připojení odpoví dříve, než bude nejstarší záznam v backlog přepsán.

Snížení SYN-RECEIVED timeout – Další možností je snížení časového limitu po který nový proces čeká ve stavu SYN-RECEIVED. Tím pádem nebudou alokované zdroje drženy tak dlouhou dobu. Bohužel to tak výrazně ovlivní i legitimní uživatele. Během probíhajícího útoku lze očekávat zvýšenou aktivitu na síti, což se může projevit ve ztrátovosti paketů. Legitimní klient tak může být odmítnut, protože potvrzení SYN/ACK nemusí vůbec dorazit ani po vícenásobném zaslání.

SYN cache – SYN cache využívá hashovací tabulku pro uchování části informací, které by se normálně ukládaly do TCB. Až ve chvíli, kdy je dokončena synchronizace, se alokuje TCB a informace z tabulky se sem přesunou.

SYN cookie – Při použití SYN cookies nový proces nepřechází do stavu SYN-RECEIVED. Místo toho jsou základní informace z TCB zkomprimovány do bitů, které tvoří sekvenční číslo v paketu SYN/ACK. Systém tedy alokuje zdroje potřebé pro komunikaci až ve chvíli, kdy přijde odpověď a potvrdí se tak legitimita druhé strany spojení. TCB je pak možné opět získat z příchozího ACK, protože sekvenční číslo bude pouze o

jedna větší a je tak možné původní TCB zrekonstruovat. Často je však problém se zakódováním celé struktury TCB do 32 bitů, což je velikost sekvenčního čísla.

Firewally a proxy servery – Jednou z možností je přenesení režie spojené se synchronizací na firewall nebo proxy server. Příkladem může být proxy server, který nepropustí synchronizační pakety až k serveru, dokud není synchronizace dokončena. Server tak není zatížen, pokud by došlo k SYN floodingu.

Tyto metody nejsou zdaleka dokonalé, a jsou určeny především pro ochranu výpočetních zdrojů serveru. DoS (DDoS) útoky tak zůstávají stále velkým problémem, protože vytvořit masivní útok, který kompletně zahltí linku není dnes vůbec složité. V takovém případě jsou výše popsané metody neúčinné.

Kapitola 4

Detekce SYN flooding útoků

V této kapitole jsou popsány tři metody z oblasti monitorování a analýzy síťového provozu, používané pro detekci SYN flooding útoků popsané v [3]. Implementace algoritmů je blíže popsána v kapitole 5.

4.1 Metoda SynFinDiff

Metoda detekce SYN flooding útoků s názvem *Synfindiff* byla poprvé představena v [11]. Autoři popisují její výhody v nízké výpočetní režii a především v bezstavovosti, což z ní dělá metodu, která je sama odolná proti SYN floodingu. Detekční mechanismus je založen na sledování dvojic TCP příznaků SYN-FIN (případně RST) během komunikace. Algoritmus patří do skupiny tzv. Sequential Change Point Detection algoritmů [2]. Ve statistické analýze jsou tyto metody využívány pro identifikaci změn ve sledovaných stochastických procesech nebo časových řadách. Aby byl mechanismus snadněji implementovatelný a obecněji aplikovatelný, využívají autoři metodu CUSUM (Cumulative Sum), která je blíže popsána v sekci 4.1.2.

4.1.1 Dvojice SYN-FIN (RST)

Algoritmus *Synfindiff* je založen především na předpokladu očekávaného výskytu TCP příznaků SYN-FIN (RST) při běžné komunikaci. Jak bylo řečeno v 3.1, každá relace TCP začíná přijetím paketu s příznakem SYN a končí přijetím paketu s příznakem FIN. Teoreticky by tedy při normálním provozu měl být počet SYN a počet FIN stejný. Při SYN flood útoku však bude počet SYN paketů mnohem větší. V réalném provozu je však v počtu SYN a FIN paketů mírná nerovnováha i za normálních podmínek. Nerovnováhu můžou vyvolat například dlouho trvající TCP spojení, jejichž ukončení pomocí FIN proběhne mimo sledovaný časový úsek a nezapočítá se do aktuální statistiky. Těchto spojení však většinou neprobíhá příliš mnoho a dopad na nerovnováhu je zanedbatelný. Hlavní příčinou této nerovnováhy je však přítomnost příznaku RST. Paket s nastaveným RST může ukončit otevřenou relaci bez nutnosti generovat FIN.

V [11] popisují dva druhy RST, které je třeba brát v úvahu v rámci *Synfindiff*:

1. Pasivní RST – RST je odesláno, pokud dorazí paket určený pro uzavřený port.
2. Aktivní RST – RST je vygenerováno, aby bylo TCP spojení přerušeno.

V zásadě však neexistuje způsob, jak je od sebe odlišit. Existují dva způsoby jak se k tomuto problému stavět. Buď klasifikovat všechny RST jako pasivní nebo naopak všechny RST jako aktivní. V prvním případě se RST nebudou započítávat do statistiky, což sníží citlivost detekčního systému. V druhém případě se RST budou započítávat jako FIN, čímž se může navýšit množství falešných poplachů.

Je tedy nutné vytvořit jakýsi vhodný poměr mezi těmito dvěma extrémy. Vzhledem k většímu množství aktivních RST než pasivních navrhují autoři algoritmu z každých čtyř RST klasifikovat tři jako aktivní a jeden jako pasivní. Aktivní RST jsou pak během měření započítány jako FIN pakety (FIN tedy označuje FIN pakety i započítané RST pakety).

4.1.2 CUSUM algoritmus

CUSUM metoda se používá k určení, zda došlo ke změně střední hodnoty μ_0 nezávislé náhodné gaussovské veličiny na $\mu_1 \neq \mu_0$. Při použití pro detekci SYN flooding útoku je pak nutné určit parametry síťového provozu, které se dají brát jako gaussovské náhodné veličiny se střední hodnotou, která je víceméně statická během normálního provozu, ale mění se, pokud dojde k útoku ([3]). Autoři metody *Synfindiff* používají jako sledovaný parametr poměr počtu SYN a FIN (RST) paketů.

Mějme $\{\Delta n, n = 0, 1, 2, ...\}$ jako rozdíl mezi SYN a FIN pakety během jedné vzorkovací periody. Protože $\{\Delta n\}$ může být závislá na denní době či konkrétním dnu v týdnu (přes noc bude například menší provoz než přes den a proto se bude její hodnota lišit), normalizujeme ji hodnotou plovoucího průměru $\bar{F}$ počtu FIN, aby byl algoritmus méně náchylný na tyto výkyvy. Hodnotu $\bar{F}$ lze získat v reálném čase a je možné ji pravidelně aktualizovat. Pro výpočet $\bar{F}$ je použita metoda Exponentially Weighted Moving Average (dále jen EWMA):

$$\bar{F}(n) = \alpha\bar{F}(n-1) + (1-\alpha)FIN(n), \tag{4.1}$$

kde n reprezentuje diskrétní index času a α je konstantní hodnota ležící striktně mezi 0 a 1.

EWMA metoda je zde použita především pro to, aby se do výpočtu $\bar{F}$ v aktuální vzorkovací periodě započítaly i všechny předešlé hodnoty $\bar{F}$, ale s klesající váhou. α je pak hodnota, která udává, jak rychle bude klesat váha předešlých $\bar{F}$. Čím je vyšší, tím rychleji klesá význam minulých pozorování.

Dále definujeme $X_n = \Delta n/\bar{F}$. Střední hodnota X_n – označme ji jako c – by měla být za normálního provozu menší než 1 a měla by být velmi blízko 0, tedy $E(X_n) = c \ll 1$. Zvolíme parametr a, který je jakousi horní hranicí X_n, $a > c$ a definujeme $\tilde{X}_n = X_n - a$ tak, aby za normálního provozu bylo $\tilde{X}_n$ záporné. Předpokládá se, že pokud dojde k útoku, $\tilde{X}_n$ poroste a bude kladné.

Mějme kumulativní součet y_n, definovaný jako

$$\begin{aligned} y_n &= (y_{n-1} + \tilde{X})^+, \\ y_0 &= 0, \end{aligned} \tag{4.2}$$

pak pokud y_n překročí nějakou předem určenou hranici N, znamená to že došlo k útoku.

Jinými slovy se dá říct, že a má za úkol snižovat případně kladné X_n tak, aby měla statistická proměnná y_n často hodnotu 0 a nenavyšovala s časem svoji hodnotu. Pokud však v několika vzorkovací periodách bude X_n nápadně velké a bude přesahovat hodnotu a, akumulovaná hodnota y_n pak překročí práh N a bude hlášen útok SYN flooding.

4.2 Metoda Synrate

Metoda *Synrate* je popsána v [10] a autoři svoji metodu přímo porovnávají s metodou *Synfindiff* z [11]. *Synrate* používá stejně jako *Synfindiff* algoritmus založený na CUSUM metodě. Místo dvojic SYN-FIN však sleduje pouze pakety s příznakem SYN a využívá EWMA výpočtu pro „předpověď" počtu SYN paketů v příští vzorkovací periodě. Výhodou oproti *Synfindiff* je především fakt, že není potřeba brát v potaz výkyvy závislé na denní době apod. Autoři uvádí dvě verze metody *Synrate*:

1. Adaptive threshold algoritmus
2. CUSUM algoritmus

4.2.1 Adaptive threshold algoritmus

Tento algoritmus je založen na testování, zda měřené hodnoty, v tomto případě počet SYN paketů, nepřesáhnou za určitý časový úsek předem určenou hranici. Abychom se vyhnuli účinkům výkyvů provozu v různých denních dobách, hranice je nastavována adaptivně na základě odhadu střední hodnoty počtu SYN paketů.

x_n značí počet paketů SYN v časovém intervalu n a $\bar{\mu}_{n-1}$ je průměrná hodnota počtu SYN získaná z měření předcházejících n. Potom podmínka pro upozornění na anomálii je

$$\text{Pokud } x_n \geq (\alpha + 1)\bar{\mu}_{n-1}, \text{pak probíhá útok v čase } n. \tag{4.3}$$

$\alpha > 0$ a představuje procentuální navýšení průměrné hodnoty, které se dá považovat za anomálii. μ_n lze spočítat buď za pomoci několika hodnot uchovaných z předešlých časových intervalů, nebo je možné použít EWMA výpočet

$$\bar{\mu}_n = \beta\bar{\mu}_{n-1} + (1 - \beta)x_n, \tag{4.4}$$

kde β představuje EWMA faktor.

Pokud je podmínka 4.3 aplikována přímo, způsobuje to velkou míru falešných poplachů, protože pokud dojde k momentálnímu výkyvu, je okamžitě nahlášen útok i pokud zrovna o útok nešlo. Z toho důvodu autoři navrhují drobnou úpravu na podmínky. Anomáli bude hlášena, pokud x_n překročí mez $(\alpha + 1)\bar{\mu}_{n-1}$ v nejméně k po sobě jdoucích intervalech. k indikuje počet po sobě jdoucích intervalů, ve kterých byla překročena hranice pro počet SYN paketů.

Tedy pokud v k za sebou jdoucích intervalech dojde k překročení počtu SYN, který byl odhadnut výpočtem EWMA ze všech předešlých sledovaných period, znamená to, že probíhá útok.

4.2.2 CUSUM algoritmus

Podobně jako v 4.1.2, je zde CUSUM metoda použita pro sledování, zda akumulovaná hodnota g_n, označovaná v 4.3 jako y_n, nepřesáhne určitou hranici h. Tedy

$$\text{Pokud } g_n \geq h, \text{pak probíhá útok v čase } n. \tag{4.5}$$

g_n lze získat následovně:

$$g_n = \left[g_{n-1} + \frac{\alpha\bar{\mu}_{n-1}}{\sigma^2}\left(x_n - \bar{\mu}_{n-1} - \frac{\alpha\bar{\mu}_{n-1}}{2}\right)\right]^+, \tag{4.6}$$

kde g_{n-1} označuje akumulovanou hodnotu všech předešlých g_n, σ^2 je rozptyl hodnot x_i a α je očekávané procentuální navýšení průměrné hodnoty $\bar{\mu}_{n-1}$ v případě, že dojde k anomálii. Stejně jako v rovnici 4.3 x_n označuje počet naměřených SYN paketů v aktuálním časovém úseku.

Problém s tímto výpočtem je především v rozptylu σ^2. Autoři neuvádí, jakým způsobem by měla být hodnota rozptylu získána. Pro výpočet rozptylu by si algoritmus musel pamatovat všechny hodnoty x_n naměřené v čase 1 až n. Tomuto by bylo lepší se vyhnout, protože čím déle by algoritmus běžel, tím více hodnot si bude třeba pamatovat. Navíc by se algoritmus měl být schopen přizpůsobit případným změnám v charakteristice provozu, čemuž výpočet ze všech předešlých hodnot příliš neodpovídá.

V tomto případě bylo tedy vhodné algoritmus upravit. Použijeme k tomu výpočet y_n z rovnice 4.3 tak, jak je použit v metodě *Synfindiff*. Na místo pevných hodnot a a N budeme používat násobky aktuálního EWMA průměru $\bar{\mu}_n$. U metody *Synfindiff* autoři navrhují aby $N = 2a$. Zde se tohoto budeme držet a stanovíme $a_n = 1,5\bar{\mu}_n$ a $N_n = 3\bar{\mu}_n$. Výsledný výpočet kumulativní hodnoty g_n tak můžeme zapsat jako

$$\begin{aligned} g_n &= (g_{n-1} + (x_n - a))^+. \\ g_0 &= 0, \end{aligned} \tag{4.7}$$

Pak pokud bude $g_n \geq N$, znamená to, že byla zjištěna anomálie v čase n.

4.3 Metoda Partial Completion Filter

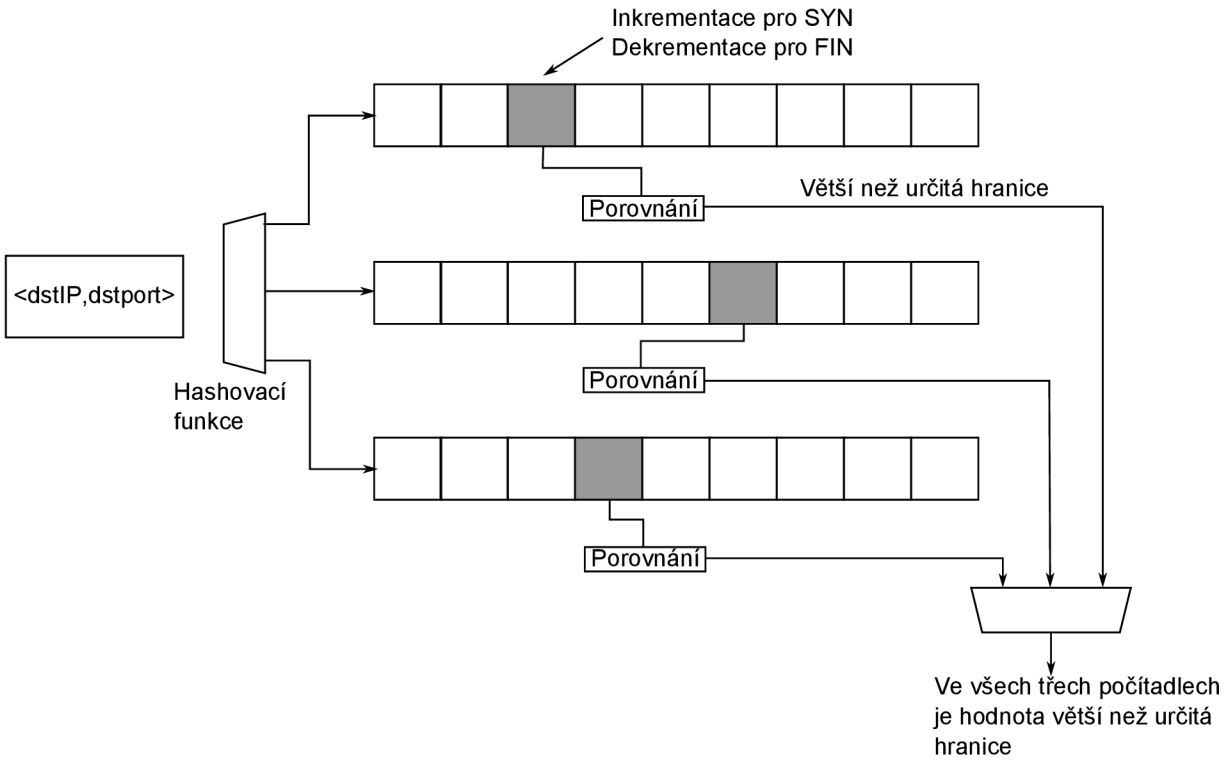
Metoda Partial Completion Filter (zkráceně PCF) byla představena v [7]. Partial completion zde označuje stav otevřené, ale neukončené TCP relace. Kromě SYN flooding útoků detekuje také tzv. útoky *Claim-and-hold*, které vytvoří velkou spoustu relací se servrem, dokončí synchronizační proces, ale ve vytvořené relaci pak nedochází k žádné aktivitě. Každý takto vytvořený proces pak musí vyčkat dokud nevyprší timeout a zabírá tak výpočetní zdroje pro legitimní připojení. PCF se tedy zaměřuje na sledování velkého počtu přijatých SYN bez ukončujících FIN paketů.

PCF používá k hashovacích tabulek označovaných jako stage. Každá tabulka je soubor počítadel. Jednotlivé tabulky používají rozdílné a navzájem nezávislé hashovací funkce pro získání hodnoty indexu ukazující na konkrétní počítadlo v tabulce. Pokud dorazí paket s nastaveným příznakem SYN, dvojice hodnot cílová IP adresa a cílový port (krátce ⟨dstIP,dstport⟩) z hlavičky paketu je použita jako hodnota pro hashovací funkci a odpovídající počítadlo v tabulce je inkrementováno. Stejně je to ve chvíli, kdy dorazí paket s příznakem FIN, počítadlo je ale naopak dekrementováno.

Algoritmus takto monitoruje provoz v za sebou jdoucích časových úsecích. Pokud dojde k tomu, že hodnoty všech počítadel, jejichž index je získán hashovacími funkcemi z jedné hodnoty ⟨dstIP,dstport⟩. přesáhne předem určenou hranici, je nahlášena anomálie. Před započetím nového časového úseku jsou počítadla vynulována.

Zde je uveden příklad pro objasnění fungování PCF. Pro účely příkladu předpokládejme že máme 3 tabulky, z nichž každá má velikost 300 položek. Řekněme, že přijde SYN paket na cílovou adresu 78.125.87.12 na port 80. Dvojice (78.125.87.12,80) tedy slouží jako hodnota pro hashovací funkce. Po zahashování třemi odlišnými funkcemi dostaneme indexy do tabulek s hodnotami například 16, 127 a 289. V první tabulce se bude inkrementovat počítadlo s indexem 16, v druhé s indexem 127 a ve třetí s indexem 289. V případě, že příchozí paket se stejnou hodnotou ⟨dstIP,dstport⟩ má příznak FIN, stejná počítadla se

budou dekrementovat. Hodnoty počítadel, jejichž index je určen stejnou vstupní hodnotou, je třeba zkontrolovat, zda nepřesáhly předem stanovenou hranici. Zkontrolujeme tedy, zda počítadla 16 v první tabulce, 127 v druhé tabulce a 289 ve třetí nejsou příliš velká. Pokud ano, znamená to výskyt anomálie.



Obrázek 4.1: Partial Completion Filter

Kapitola 5

Implementace

5.1 Nemea framework

Metody pro detekci SYN flooding byly implementovány na systému Nemea [1], který je vyvíjen v rámci organizace CESNET. Nemea (Network Measurements Analysis) je framework, který umožňuje poskládat z jednotlivých modulů, které jsou v rámci frameworku vyvíjeny, systém, jehož úkolem je automaticky analyzovat nebo zpracovávat datové toky sledované v reálném čase. Jednotlivé moduly jsou nezávisle pracující jednotky, navzájem propojené pomocí rozhraní. V zásadě každý modul pracuje s tokem dat na vstupním rozhraní, data nějakým způsobem zpracuje nebo analyzuje a posílá nový tok dat na svoje výstupní rozhraní.

Moduly je možné přidávat a odebírat dynamicky. Pokud je modul napojen na rozhraní jiného modulu, které zatím není k dispozici (například druhý modul není prozatím spuštěn), modul se periodicky snaží připojit, dokud není rozhraní aktivní. Tímto způsobem je možné přidávat nebo odebírat moduly za běhu s žádným nebo pouze s minimálním dopadem na celý systém. Pro propojení modulů bylo vyvinuto Traffic Analysis Platform (dále jen TRAP) API, které je součástí sdílené knihovny nazvané libtrap.

Sdílená knihovna libtrap, která je používaná každým implementovaným modulem, vytváří pro ulehčení propojení modulů jednoduché API. Díky tomuto API je potřeba u každého modulu specifikovat pouze jeho vstupní a výstupní rozhraní, samotné propojení nebo způsob výměny dat má pak na starosti právě libtrap. Knihovna je složena z několika základních vrstev.

Na nejnižší vrstvě jsou TCP socket pro vzdálenou komunikaci a UNIX socket pro lokální komunikaci mezi rozhraními. Dále je v rámci libtrap vyvinuta bufferovací podvrstva, která agreguje požadavky jednotlivých rozhraní, čímž efektivně snižuje režii spojenou s transferem dat a umožňuje tak zpracovávat velké objemy dat ve vysokých rychlostech. Aby mohl každý modul ovládat nižší vrstvy libtrap, jsou zde k dispozici funkce Auto-Flush, TimeOut a Multi-Read. Auto-Flush umožňuje vyprázdnit buffer, aby nedocházelo ke stárnutí nashromážděných datových toků. TimeOut definuje několik základních časovačů, které jsou jednotlivými moduly využívány pro stanovení časových prodlev při volání funkcí send nebo receive. Pro tyto účely existují tři časovače: TRAP_WAIT, TRAP_NO_WAIT a TRAP_HALF_WAIT. V zásadě se jedná o určení, zda se budou funkce recieve a send na vstupních a výstupních rozhraních chovat jako blokující či neblokující. Poslední funkcí je Multi-Read, který modulům umožňuje číst data ze všech definovaných vstupních rozhraní paralelně. Na nejvyšší vrstvě je samotné TRAP API zpřístupňující modulům nižší vrstvy libtrap.

Data jsou zasílána mezi rozhraními v podobě datových záznamů. Tyto záznamy jsou zpracovávány tzv. stream-wise, tedy jeden po druhém hned, jak jsou přijímány, bez ukládání a dělení do časových intervalů (pokud ovšem konkrétní modul explicitně nevyžaduje jiný přístup). Tento přístup má několik výhod. Jednou z nich je především fakt, že pokud jsou data zpracovávána v reálném čase, nevzniká tak zpoždění v analýze přijatých dat. Navíc není potřeba data ukládat na pevný disk, data jsou zpracována přímo v paměti. Pokud je však potřeba data uložit pro pozdější analýzu, i tato možnost je v systému k dispozici.

Všechny datové toky mezi dvěma přímo propojenými rozhraními musejí mít stejný formát, což znamená že obsahují stejné položky. Různá rozhraní však mohou specifikovat, s jakým formátem budou pracovat. Formát datového toku na vstupním či výstupním rozhraní je opět udáván dynamicky při zavádění nového modulu do systému. Díky této dynamičnosti je zároveň možné do specifikací dat přidávat nové položky, které bude využívat nově implementovaný modul, aniž by to nějakým způsobem ovlivnilo funkcionalitu již zavedených modulů. Protokol, který specifikuje, jak definovat tyto záznamy, se nazývá UniRec (Unified Record).

UniRec záznam je specifický formát dat, používaný pro komunikaci přes TRAP rozhraní. Každý záznam se skládá z několika položek, z nichž každá má jméno a typ. Soubor položek v záznamu se nazývá *template*. Záznamy jsou ukládány podobně jako struktury jazyka C, jednotlivé položky jsou uloženy za sebou. Záznam se typicky skládá ze statických a dynamických položek. Statické položky mají pevnou velikost, dynamické pak umožňují vytvářet položky s proměnnou velikostí a jsou uloženy na konci záznamu. Ukazatele na konce jednotlivých dynamických položek jsou uloženy na konci statické části záznamu. Jednotlivé položky mohou mít jakýkoli datový typ definovaný v jazyce C, navíc je možné využít řetězec znaků anebo jeden ze dvou speciálních datových typů - datový typ pro uložení tzv. timestamp (časový údaj začátku nebo konce datového toku) a datová struktura pro ukládání IPv4 a IPv6 adres.

5.2 Moduly

Moduly *Synfindiff* a *Synrate* byly implementovány v programovacím jazyce C. Modul PCF byl implementován v jazyce C++. Moduly využívají knihovny trap.h, ve které jsou definovány základní makra, konstanty a proměnné potřebné k propojení modulů s libtrap knihovnou a knihovnu unirec.h, která je potřeba pro práci s UniRec záznamy v rámci systému Nemea. Moduly v zásadě pracují se dvěma základními informacemi z přijatých UniRec záznamů. Jsou to TCP_FLAGS a TIME_LAST.

TCP_FLAGS je 8 bitový integer reprezentující hodnotu příznaků v hlavičce TCP. Pomocí bitové operace *and* je pak zjištěno, zda se v konkrétním datovém toku vyskytly příznaky SYN nebo FIN. Zde by mohl nastat problém, protože obecně datový tok může trvat delší dobu a pakety s různými příznaky se v něm mohou vyskytnout vícekrát, přičemž datové toky neuchovávají informace o počtu paketů se stejným příznakem, pouze indikují, že se takový paket objevil někdy v průběhu trvání toku. V případě SYN a FIN to však nevadí, protože každý z nich by se v jednom konkrétním toku měl objevit pouze jednou.

TIME_LAST označuje časový údaj konce datového toku ve formátu Unix timestamp. Pro statistické účely je koncový čas vhodnější, protože v UniRec záznamu jsou pak obsaženy všechny informace o konkrétním toku. Problém se zpracováním času datových toků je především v tom, že různé toky mají různé délky trvání a obecně nekončí postupně za sebou. V implementovaných modulech je tento problém řešen následovně:

- V případě prvního zpracovaného datového toku se začátek zaokrouhlí směrem dolů podle délky časového okna. Pokud tedy máme např. 5 minut (300 sekund) dlouhé časové okno, počáteční čas monitorování provozu bude určen jako `begin_time = TIME_LAST - (TIME_LAST % 300)`.
- V případě, že je zaznamenán datový tok, který patří do předešlé vzorkovací periody, takový tok se nezapočítá do aktuální statistiky.
- V případě, že je zaznamenán datový tok, který patří do aktuální periody, je započítán.
- V případě, že je zaznamenán datový tok, jehož čas je větší než `begin_time + 300`, započítají se statistické hodnoty z aktuální periody, statistické proměnné se vynulují a počáteční čas vzorkovací periody se posune o velikost časového okna na `begin_time += 300`.

Všechny algoritmy využívají pro příjímání dat časovač TRAP_WAIT, popsaný v sekci 5.1, což znamená, že pracují v blokovacím režimu. Jádrem všech algoritmů je jednoduchý `while` cyklus. Během každého cyklu je zpracován jeden UniRec záznam přijatý na vstupní rozhraní modulu. Pokud algoritmus detekuje anomálii, reaguje odesláním upozornění na své výstupní rozhraní.

5.2.1 Synfindiff

Jak je popsáno v sekci 4.1, algoritmus monitoruje množství SYN a FIN v příchozích datech. Z počtu SYN a FIN jsou pak získávány hodnoty Δn a $\bar{F}$. Zatímco δn se v každé vzorkovací periodě získá jednoduše jako SYN - FIN, $\bar{F}$ by bylo hned v první periodě metodou EWMA vypočítáno pouze z jedné hodnoty počtu FIN, proto algoritmus několik prvních časových úseků pouze zaznamenává počet SYN a FIN, ze kterých vypočítává $\bar{F}$ a tzv. se „učí“. Po této učební době se vypočtené $\bar{F}$ při každé změně času monitorovacího okna aktualizuje a je použito pro výpočet y_n. y_n je sčítáno s hodnotami z předchozích period a pokud přesáhne zadaný threshold, na výstupní rozhraní je zaslán timestamp s časem začátku aktuální periody a číselný údaj indikující SYN flooding útok.

Hlavní parametry ovlivňující průběh algoritmu jsou především α ve výpočtu EWMA průměru $\bar{F}$ a hodnoty a a N. V sekci 4.1 jsou posány důvody pro nerovnováhu v počtu SYN a FIN. Tato nerovnováha hraje roli především při určování hodnot a a N a v podstatě je vyjádřena hodnotou y_n. Autoři algoritmu uvádí, že by se hodnota y_n měla v ideálním případě blížit nule. Algoritmus má v souboru synfindiff.h nastaveny všechny parametry na předem zvolené hodnoty, v kapitole 6 je však ukázáno, že parametry je vhodné upravit podle potřeb dané situace a proto je možné všechny hlavní parametry při spuštění explicitně změnit. Kromě výše zmíněných α, a a N lze také změnit délku vzorkovací periody. Jakým způsobem parametr α a délka časového okna ovlivňují průběh monitorování, je znázorněno v kapitole 6.

5.2.2 SynRate

Metoda *Synrate* je velmi podobná v implementaci na metodu *Synfindiff*. Zde se monitoruje pouze počet SYN. Z počtu SYN z předešlých period se pomocí EWMA výpočtu „předpovídá“, jaké množství SYN by se mělo objevit v aktuální periodě. Jak přesná je předpověď počtu SYN, závisí především na EWMA faktoru β z rovnice 4.4.

V sekci 4.2 bylo řečeno, že parametry a a N jsou implicitně nastaveny na hodnoty 1,5 a 3. Tyto parametry je spolu s β a s délkou vzorkovací periody možné explicitně změnit. V kapitole 6 je ukázáno, jakým způsobem je parametrem β předpověď EWMA ovlivněna.

Pokud je detekován útok, algoritmus si uchová poslední hodnotu EWMA předpovědi, dokud není jisté, že útok pominul. Jinak by se totiž snažil hodnotu EWMA aktualizovat i během probíhajícího útoku a tím by do předpovědi zanesl nesprávná data. Ve chvíli, kdy konečně počet SYN klesne pod hranici hodnoty threshold, je tato hodnota opět započítána do výpočtu EWMY.

5.2.3 PCF

Metoda PCF popsaná v sekci 4.3 podobně jako předešlé metody monitoruje počet SYN a FIN. Z přijatého UniRec záznamu jsou v tomto případě potřeba navíc hodnoty DST_IP (cílová IP adresa) a DST_PORT (cílový port). Z kombinace těchto dvou hodnot je vytvořen řetězec pro tři různé hashovací funkce, které z nich vytvoří číselné hodnoty reprezentující indexy do tří hashovacích tabulek (stage). Každá hashovací tabulka je implementována jako pole typu `long int`. Zde je potřeba počítat s tím, že jednomu indexu v poli může odpovídat více než jedna hodnota řetězce ⟨dstIP,dstport⟩. Proto je potřeba zvolit jakýsi kompromis mezi počtem možných hodnot ⟨dstIP,dstport⟩ a velikostí jednotlivých polí. Během monitorování provozu z monitorovacích sond bylo zjištěno průměrně 800 tis. unikátních hodnot ⟨dstIP,dstport⟩ během period dlouhých 300 sekund. Velikost jednoho pole je proto implicitně nastavena na hodnotu 100 tis.

Aby bylo možné při detekování anomálie určit na jakou IP adresu a port byl útok veden (zdrojové IP adresy nemá příliš význam zjišťovat, protože ve většině případů se jedná o spoofované hodnoty), je každá nová hodnota ⟨dstIP,dstport⟩ uložena do datové struktury `set<string>`. Navíc jsou hodnoty z této struktury později použity pro zjištění indexů do jednotlivých hashovacích tabulek.

Na konci každého časového úseku jsou zkontrolovány všechny odpovídající trojice počítadel ve všech hashovacích tabulkách. Protože by se kvůli této kontrole musel algoritmus zastavit, dokud by nebyly zkontrolovány všechny položky, běží modul ve více vláknech. Pro kontrolu počítadel v hashovacích tabulkách se pomocí volání `std::thread.detach()` spustí samostatné vlákno, kterému jsou předány hashovací tabulky a struktura uložených ⟨dstIP,dstport⟩. Takto může hlavní vlákno okamžitě pokračovat v monitorování síťového provozu.

Aby mělo hlavní vlákno k dispozici prázdné hashovací tabulky a strukturu řetězců, zatímco oddělené vlákno pracuje s hodnotami v těch předešlých, jsou na začátku vytvořeny dvě sady polí (celkem tedy šest) a dvě struktury řetězců. Na tři pole a jednu strukturu ukazují ukazatele. Tyto ukazatele jsou použity pro naplnění struktur hodnotami získanými během monitorování. Novému vláknu jsou tedy předány čtyři ukazatele a jakmile se vlákno oddělí, struktury referencované ukazateli se prohodí. S původními strukturami teď pracuje samostatné vlákno a s druhou sadou datových struktur pracuje hlavní vlákno.

Samostatné vlákno postupně prochází hodnoty ze struktury řetězců a z každé z nich získá tři indexy do hashovacích tabulek. Pomocí těchto indexů získá z tabulek tři číselné hodnoty označující rozdíl počtu SYN a FIN. Pokud jsou všechny tři hodnoty zároveň vyšší než pevně stanovený threshold, na výstupní rozhraní se odešle oznámení o anomálii a informace o cílové IP a cílovém portu, na které útok probíhal.

Při spuštění modulu je možné explicitně měnit délku vzorkovací periody, velikost hashovací tabulky a především threshold. Jak bylo řečeno výše, implicitní velikost hashovací

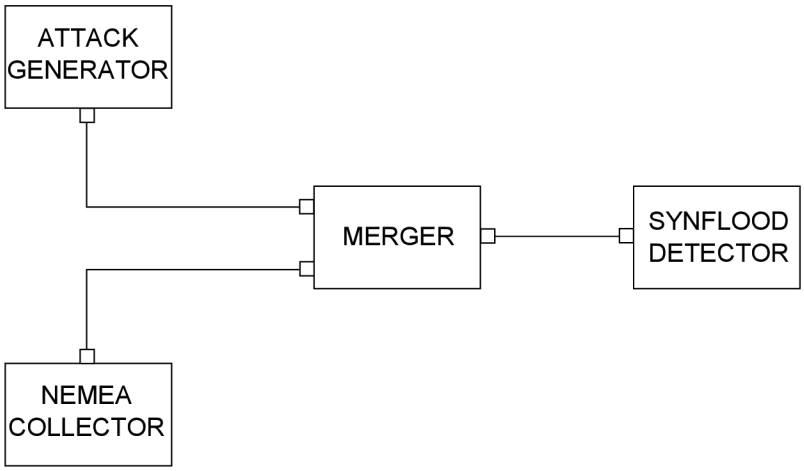
tabulky byla stanoven na 100 tis. a délka priody je 300 s.

Určení thresholdu je složitější, protože se více řetězců ⟨dstIP,dstport⟩ může zahashovat na jeden číselný index do pole (viz. výše). Proto se může stát, že se více datových toků z různých IP adres agreguje do jednoho počítadla v hashovací tabulce. Pokud by tedy byl threshold příliš nízký, vyústí to ve velkou míru falešných poplachů i v případě, že se nejedná o útok. Na druhou stranu pokud by byl příliš vysoký, může se stát, že SYN flooding útok nebude detekován. Během monitorování sítě za běžného provozu byly nejvyšší hodnoty v jednotlivých počítadlech průměrně mezi 30–50 tis.

Kapitola 6

Testování

Jakým způsobem probíhalo testování modulů, je zobrazeno na obr. 6. Hlavním modulem je modul s názvem merger. Merger má obecně n vstupních rozhraní a jejich datové toky agreguje do jednoho výstupního. Zde jsou na merger napojena dvě vstupní rozhraní. Jedno vstupní rozhraní je připojeno na monitorovací sondy (sjednoceny pod názvem collector nemea) a přichází přes něj data přímo v reálném čase. Na druhé rozhraní je připojen modul studenta Dávida Filička [6], který je implementován za učelem generování síťových útoků. Data z reálného provozu a z generátoru útoků se tak pomocí modulu merger agregují a jsou poslána na výstupní rozhraní. Na něj je pak napojen některý z modulů detekujících SYN flooding.



Obrázek 6.1: Zapojení modulů pro testování v rámci systému Nemea

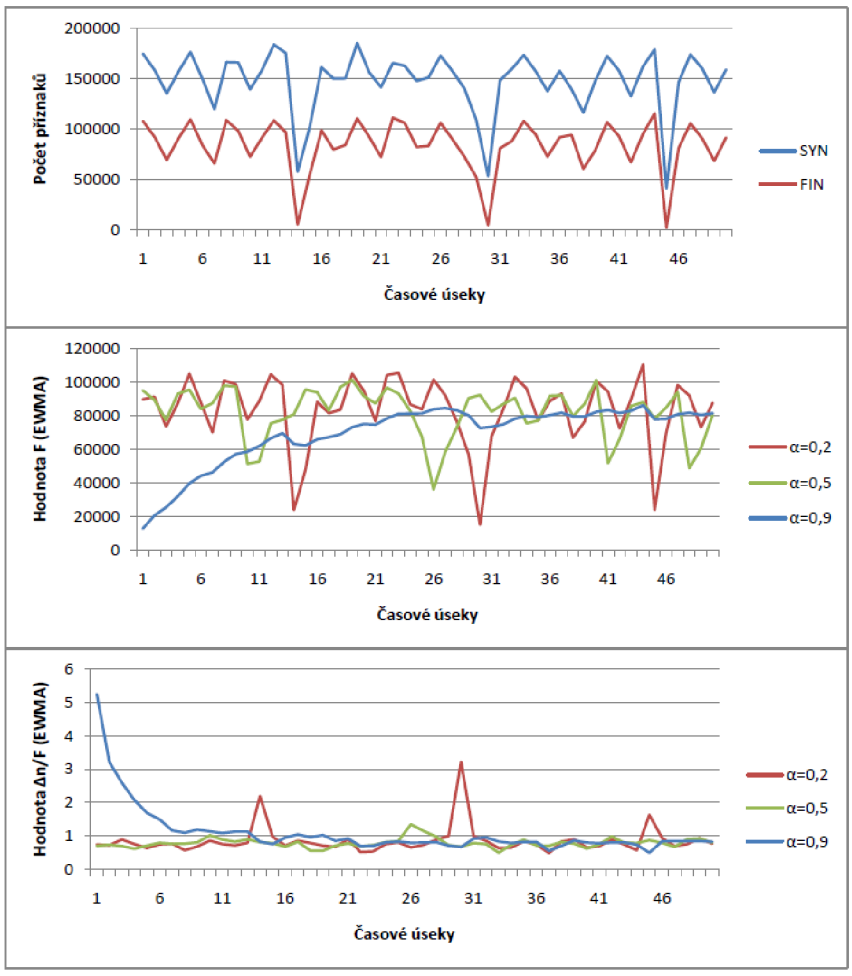
6.1 Synfindiff

Trojice grafů na obrázku 6.2 popisuje, jakým způsobem se budou chovat důležité hodnoty algoritmu při odlišných nastaveních parametru α. Test byl proveden s nastavením délky okna na 40 sekund. Horní obrázek zobrazuje počet naměřených SYN a FIN v jednotlivých

časových úsecích. Test byl proveden na stejné sadě dat, takže počet SYN a FIN je pro oba dva spodní grafy stejný.

Na prostředním grafu je vidět chování hodnoty $\bar{F}$ při nastavení parametru α na hodnoty 0,2 a 0,9. Je zde patrné, že při nastavení 0,2 se hodnota $\bar{F}$ velmi rychle přizpůsobuje změnám v počtu SYN a FIN. Na druhou stranu při nastavení hodnoty na 0,9 křivka lépe „vyhlazuje“ hodnotu $\bar{F}$ a vytváří tak lepší obecný model chování.

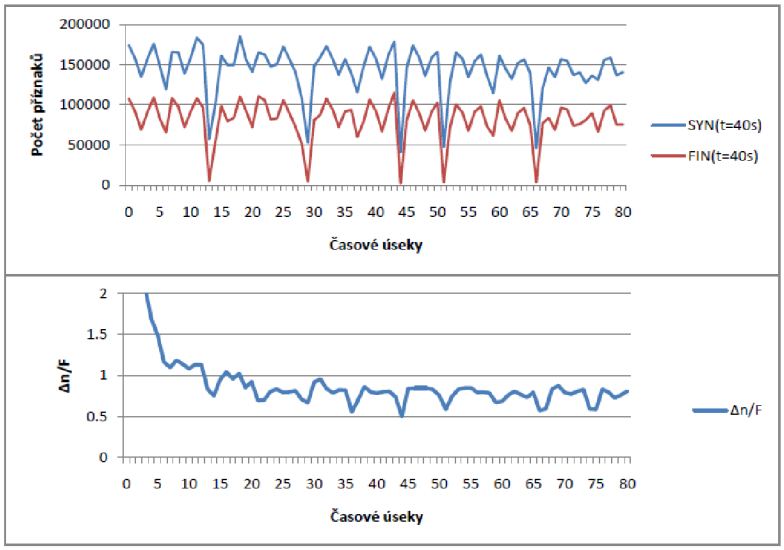
Na spodním grafu je vidět vývoj hodnoty $\delta n/\bar{F}$. Pokud je parametr α nastaven na hodnotu 0,2, počáteční učební doba je velmi krátká, opět se však hodnota až příliš rychle přizpůsobuje změnám v počtu SYN/FIN. Při nastavení parametru na 0,9 je patrná delší doba učení na začátku, algortimus však mírněji reaguje na změny a opět lépe vyjadřuje model chování v delším čase. Z těchto testů je zřejmé, že ideální nastavení parametru α pro algoritmus Synfindiff je hodnota co nejblíže jedné.



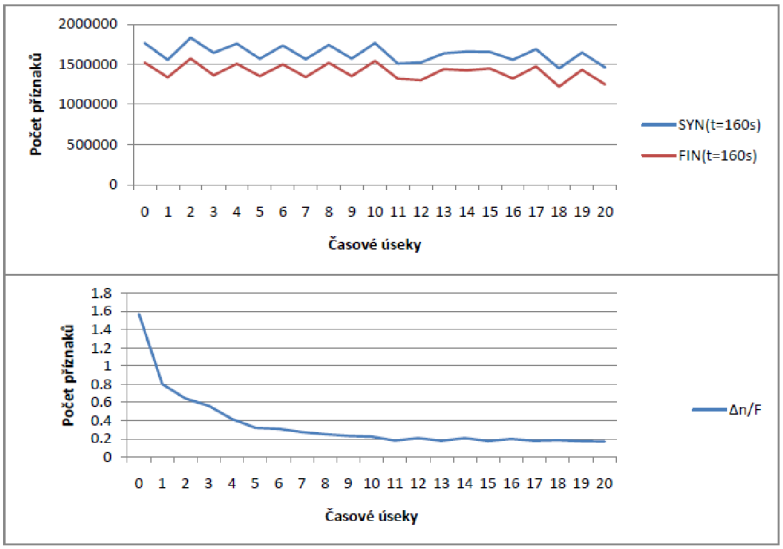
Obrázek 6.2: Porovnání nastavení parametru α u metody Synfindiff

Vzhledem k povaze chování protokolu TCP by bylo dobré zde zmínit, jakým způsobem může monitorování ovlivnit délka monitorovacího okna. Protože TCP obecně nemá určenou nějakou maximální délku relace, dá se očekávat, že se FIN jedné relace neobjeví ve stejném

okně jako SYN. Tento fakt ovlivní vypočítanou hodnotu $\delta n/\bar{F}$ a tím pádem je potřeba správně nastavit a a N. Parametr α při testu byl vzhledem k předešlým testům nastaven na hodnotu 0,9. Dvojice grafů 6.3 ukazují počet SYN/FIN a hodnotu $\delta n/\bar{F}$ při délce okna 40 sekund. Grafy 6.4 pak znázorňují to samé při délce okna 160 sekund. Každá dvojice grafů sice ukazuje jiný počet časových oken, ve výsledku však ukazují stejný časový úsek (80 oken po 40s je 3200s, 20 oken po 160s je také 3200s).



Obrázek 6.3: Délka monitorovacího okna nastavena na 40s

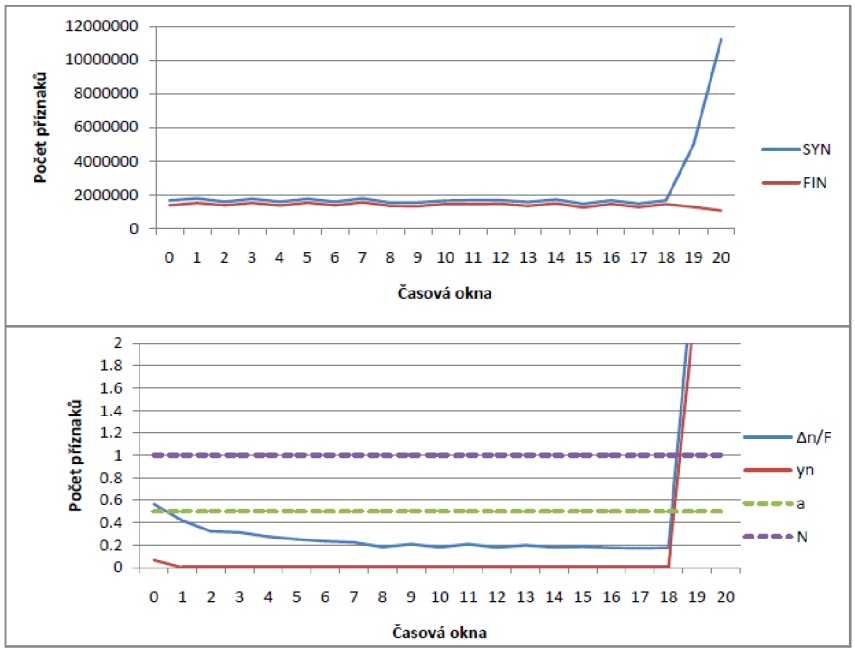


Obrázek 6.4: Délka monitorovacího okna nastavena na 160s

V dvojici grafů na obrázku 6.3 je vidět, že poté, co se algoritmus ustálí, se hodnota $\delta n/\bar{F}$ pohybuje mezi 0,6 a 0,8. To by znamenalo, že nastavení parametru a by muselo být na hodnotu alespoň 0,8, možná dokonce 1. V dvojici grafů na obrázku 6.4 se hodnota $\delta n/\bar{F}$ blíží spíše hodnotě 0,2. a by tedy stačilo nastavit například na hodnotu 0,4. Z těchto dat

tedy vyplývá, že by pak bylo vhodné monitorovat provoz ještě před konečným spuštěním algoritmu, aby se správně nastavily hodnoty a a N.

Na konec následuje ukázka algoritmu při detekci SYN flood útoku. Nastavení hlavních parametrů algoritmu bylo 160s pro délku okna, $a = 0,5$, $N = 1$ a $\alpha = 0,9$. Na obrázku 6.5 je vidět vývoj hodnot $\delta n/\bar{F}$ a y_n při útoku.



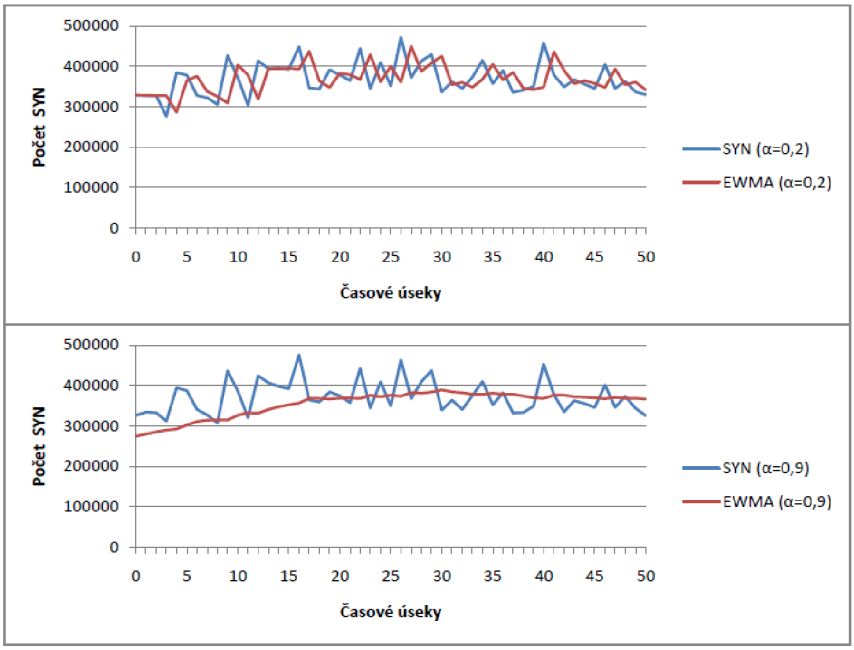
Obrázek 6.5: Ukázka monitorování algoritmem Synfindiff při útoku SYN flood

6.2 Synrate

Graf 6.6 zobrazuje, jakým způsobem je „předpověď“ počtu SYN v příští vzorkovací periodě ovlivněna parametrem β u EWMA výpočtu. V prvním případě je parametr β nastaven na hodnotu 0,2 a v druhém případě na hodnotu 0,9, přičemž délka časového okna je nastavena na 60 sekund. V horním grafu je vidět, že pokud je parametr β nastaven na hodnotu blíže k nule, EWMA předpověď poměrně věrně kopíruje tvar křivky znázorňující počet SYN. Na spodním grafu je vidět, že pokud je parametr β nastaven blíže k jedničce, EWMA předpověď lépe „vyhlazuje“ případné odchylky a výsledná křivka mnohem lépe charakterizuje dlouhodobý model chování sledovaného provozu.

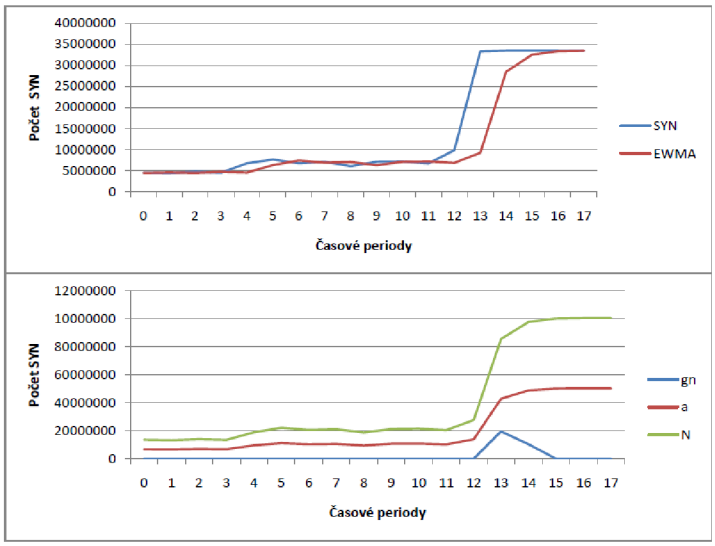
Grafy 6.7 a 6.8 zobrazují, jakým způsobem se chová akumulovaná hodnota g_n v případě útoku. První dvojice grafů znázorňuje reakci na útok při nastavení parametru β na hodnotu 0,2, druhý graf ukazuje reakci při $\beta = 0,9$, délka periody je 60s. Zde je jasně vidět, že při nastavení $\beta = 0,2$ se algoritmus až příliš rychle přizpůsobí. g_n sice zaznamená změnu v provozu, ale jen na přibližně dvě periody. Po nich se algoritmus přizpůsobí množství SYN, které jsou zaznamenány během útoku, g_n nikdy nepřekročí horní hranici a útok zůstane bez povšimnutí.

Na druhé dvojici grafů je vidět, že pokud je parametr $\beta = 0,9$, pak při zahájení útoku

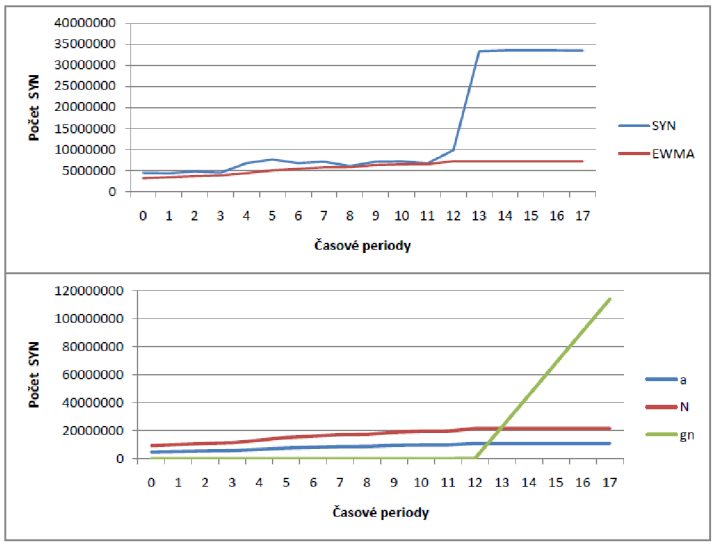


Obrázek 6.6: Hodnoty SYN a EWMA algoritmu Synrate s různým nastavením parametru β

g_n okamžitě reaguje a během několika málo period překročí horní hranici. EWMA se po zaregistrování útoku přestane zvyšovat, protože „zamrzne“ na hodnotě posledního výpočtu před detekováním útoku. Z této ukázky zcela jasně vyplývá, že parametr β by měl být nastaven na hodnotu blízko jedné.



Obrázek 6.7: Synrate při nastavení $\beta = 0,2$



Obrázek 6.8: Synrate při nastavení $\beta = 0,9$

6.3 PCF

Algoritmus PCF je od prvních dvou metod velmi odlišný. Nevytváří žádný model, ani statistiku provozu sítě, pouze sleduje jestli na jednu konkrétní adresu není zasláno příliš mnoho SYN a málo FIN. Jeho jediným parametrem, který je třeba nastavit, je threshold. Ten je možné nastavit na poměrně vysokou hodnotu, protože pokud dojde k útoku na jednu konkrétní adresu a port, jen ta jedna se pak v počítadle projeví. V sekci 5.2.3 je uvedeno, že po monitorování sítě byla hodnota nastavena na 100 tis. Pomocí parametrů zadaných při spuštění je možné hodnotu explicitně přenastavit.

6.4 Shrnutí

Po testech, určených ke zjištění ideálního nastavení parametrů monitorovacích algoritmů, bylo vyzkoušeno několik útoků. Při všech testech byly útoky úspěšně detekovány. Ukázky jsou uvedeny výše. PCF algoritmus byl schopen kromě výskytu útoku zjitit také cílovou IP adresu a port, na kterou byl útok veden.

Kapitola 7

Závěr

V rámci této práce byly implementovány tři metody pro detekci anomálií v síťovém provozu, konkrétně pro odhalení útoku typu SYN flood. Všechny metody byly otestovány na reálných datech z monitorovacích sond organizace CESNET. Všechny popsané metody byly původně vytvořeny pro monitorování siťového provozu na bázi per-paket, pro účely této práce však byly použity pro monitorování na bázi per-flow. Tento přístup je možný především díky povaze chování protokolu TCP. Všechny metody byly implementovány tak, jak je jejich autoři popisují, v rámci každého algoritmu však byly provedeny jisté úpravy.

Zásadní úprava byla provedena především u metody *Synrate*. V zásadě byla snaha, vyhnout se složitému počítání rozptylu naměřených hodnot, kdy by si algoritmus musel v čase n pamatovat všechny hodnoty naměřené v čase 1 až n. Část metody pro předpověď počtu SYN paketů v dalším časovém úseku byla ponechána, výpočet akumulované hodnoty, která je použita pro určení, zda se v provozu nevyskytla anomálie, byl upraven podle metody *Synfindiff*. Threshold, který pokud je překročen akumulovanou hodnotou, indikuje výskyt anomálie, není určen napevno jako u metody *Synfindiff*, ale je v každém časovém okně vypočítán jako určitý násobek předpovědi počtu SYN paketů. Tento přístup vykazuje lepší flexibilitu a přizpůsobivost algoritmu na případné legitimní navýšení množství provozu v síti.

U obou metod *Synfindiff* a *Synrate* je velmi detailně popsáno, jak určit časový okamžik, kdy došlo k výskytu anomálie. Bohužel však autoři metod neřeší, jak určit časový okamžik, kdy útok skončil a nenavrhují žádný postup, jak by se algoritmus po detekci útoku měl vrátit do vychozího klidového stavu. Metoda *Synfindiff* byla proto upravena tak, že akumulovaná hodnota, která by při dostatečně dlouhém útoku rostla pořád dál, má určenou jistou maximální hodnotu, které když dosáhne, dále se nezvyšuje. Pokud se provoz na síti uklidní, umožní to algoritmu navrátit se do stavu jako před útokem. U metody *Synrate* byl postup stejný, navíc je zde implementováno „zamrznutí“ předpovědi počtu SYN paketů v případě detekování narušení. Hodnota zůstane taková, jaká byla v posledním časovém úseku před detekcí útoku, aby po skončení útoku nepoužíval ve statistice vysoké hodnoty z časových úseků, kdy probíhal útok.

Metoda PCF je implementována vícevláknově, aby se tak urychlil celkový běh algoritmu. Zatímco hlavní vlákno monitoruje síťový provoz, vedlejší vlákno provádí kontrolu naměřených hodnot z předešlého časového úseku. Algoritmus si během monitorování uchovává hodnoty všech cílových IP adres a portů, na které přichází komunikace, aby bylo možné při detekování anomálie určit, na kterou adresu a port byl útok veden. Algoritmus uchovává pouze cílové údaje, protože u zdrojových se dá předpokládat, že budou podrvžené.

Literatura

[1] Bartoš, V.; Ž., M.; Č., T.: *Nemea: Framework for stream-wise analysis of network traffic*. CESNET, Praha, Czech Republic, 12 2013.

[2] Basseville, M.; Nikiforov, I. V.: *Detection of Abrupt Changes: Theory and Application*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993, ISBN 0-13-126780-9.

[3] Beaumont-Gay, M.: A Comparison of SYN Flood Detection Algorithms. In *Proceedings of the Second International Conference on Internet Monitoring and Protection*, ICIMP '07, Washington, DC, USA: IEEE Computer Society, 2007, ISBN 0-7695-2911-9.

[4] Eddy, W.: TCP SYN Flooding Attacks and Common Mitigations. RFC 4987 (Informational), Srpen 2007.

[5] Eddy, W. M.: Defenses Against TCP SYN Flooding Attacks. *The Internet Protocol Journal*, ročník 9, č. 4, December 2006: s. 2–16.

[6] Filičko, D.: Simulátor síťových útoků. 2014.

[7] Kompella, R. R.; Singh, S.; Varghese, G.: On Scalable Attack Detection in the Network. *IEEE/ACM Trans. Netw.*, ročník 15, č. 1, Únor 2007: s. 14–25, ISSN 1063-6692.

[8] Patcha, A.; Park, J.-M.: An Overview of Anomaly Detection Techniques: Existing Solutions and Latest Technological Trends. *Comput. Netw.*, ročník 51, č. 12, Srpen 2007: s. 3448–3470, ISSN 1389-1286.

[9] Postel, J.: Transmission Control Protocol. RFC 793 (INTERNET STANDARD), Září 1981, updated by RFCs 1122, 3168, 6093, 6528.

[10] Siris, V. A.; Papagalou, F.: Application of Anomaly Detection Algorithms for Detecting SYN Flooding Attacks. *Comput. Commun.*, ročník 29, č. 9, Květen 2006: s. 1433–1442, ISSN 0140-3664.

[11] Wang, H.; Zhang, D.; Shin, K.: Detecting SYN flooding attacks. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, ročník 3, June 2002, ISSN 0743-166X, s. 1530–1539.

[12] Zhang, W.; Yang, Q.; Geng, Y.: A Survey of Anomaly Detection Methods in Networks. In *Computer Network and Multimedia Technology, 2009. CNMT 2009. International Symposium on*, Jan 2009, s. 1–3.

Dodatek A

Obsah CD

- tex/ – zdrojový kód písemné zprávy
- synflood/synfindiff/ – zdrojové kódy modulu *Synfindiff*
- synflood/synrate/ – zdrojové kódy modulu *Synrate*
- synflood/PCF – zdrojové kódy modulu *PCF*
- readme.txt – návod na instalaci a použití modulů