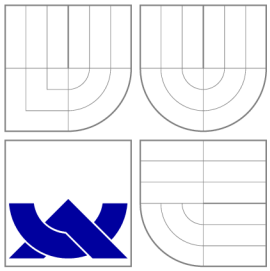
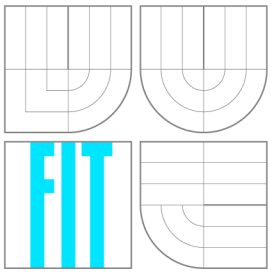


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

GENEROVÁNÍ A OCHRANA PROTI DOS ÚTOKU NA TRANSPORTNÍ VRSTVĚ

TRANSPORT LAYER DOS ATTACK GENERATOR AND PROTECTION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — LUKÁŠ PELÁNEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR MUSIL
SUPERVISOR

BRNO 2016

Zadání bakalářské práce/18703/2015/xpelan03

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Pelánek Lukáš**
Obor: Informační technologie
Téma: **Generování a ochrana proti DOS útoku na transportní vrstvě**
Transport Layer DOS Attack Generator and Protection
Kategorie: Počítačové sítě

Pokyny:

1. Prostudujte dostupnou literaturu týkající se DOS útoků. Zaměřte se blíže na útoky na transportní vrstvu.
2. Seznamte se knihovnou Pcap a s možnostmi generování síťových dat.
3. Navrhněte aplikaci umožňující generování DOS útoků.
4. Aplikaci implementujte a otestujte.
5. Diskutujte možnosti ochrany proti implementovaným útokům.
6. Zhodnoťte výsledky práce a diskutujte případné pokračování nebo rozšíření práce.

Literatura:

- Dle pokynů vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2 zadání

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Musil Petr, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato bakalářská práce se zaměřuje na problematiku síťových útoků Denial of Service a obranu proti těmto útokům. V práci je vysvětleno fungování počítačových sítí, principy útoků DoS a obrana. Druhá část práce se zabývá návrhem a implementací aplikace, která dokáže generovat síťové útoky SYN flood, UDP flood a ICMP flood. V závěru práce je popsán průběh testování aplikace a zhodnocení dosažených výsledků.

Abstract

This bachelor thesis focuses on the issues of Denial of Service attacks and the defense against them. It explains the inner workings of computer networks, the principles of DoS attacks and the defense against them. The second part of the thesis focuses on the design and deployment of an application that is able to generate network attacks SYN flood, UDP flood and ICMP flood. The conclusion of this thesis describes the process of testing the application and evaluation of the achieved results.

Klíčová slova

DoS, generování, ochrana, pcap, SYN flood, UDP flood, ICMP flood

Keywords

DoS, generator, protection, pcap, SYN flood, UDP flood, ICMP flood

Citace

PELÁNEK, Lukáš. *Generování a ochrana proti DOS útoku na transportní vrstvě*. Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Musil Petr.

Generování a ochrana proti DOS útoku na transportní vrstvě

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Petra Musila. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Lukáš Pelánek
18. května 2016

Poděkování

Děkuji svému vedoucímu Ing. Petru Musilovi za odborné vedení a komentáře, které mi při řešení této bakalářské práce poskytl.

Obsah

Kapitola 1

Úvod

Žijeme v době, kdy nás informatika obklopuje ze všech stran. Už od samotného počátku počítačových sítí a internetu bylo důležité počítat s hrozbou síťových útoků. Prakticky denně přicházíme do kontaktu s počítačovými sítěmi a drtivá většina z nás se s kybernetickými útoky již setkala, ač si to ani nemusíme uvědomovat. Síťové útoky typu Denial of Service (DoS) jsou aktuální tématikou a především problematikou, protože proti těmto typům útoků neexistuje žádná spolehlivá ochrana. K DoS útokům dochází ve světě prakticky každou chvíli a útočníkem se může stát i naprostý amatér, síťový expert a v nejhorším případě také organizovaná skupina s jednotným cílem a zaměřením. Právě organizované skupiny útočníků tvoří ve světě kybernetických útoků největší problémy a způsobují největší škody.

Ve světě počítačových sítí existuje široké spektrum útoků s různým zaměřením. Některé útoky mají za cíl vyřadit určenou službu z provozu, další získat osobní informace, zamezit uživateli veškerou elektronickou komunikaci s okolním světem nebo vyjádřit nesouhlas s určitým postojem nebo výroky. Tyto útoky jsou velice nebezpečné a většinou míří na významná místa. Například bankovní společnosti, politické strany či zpradovajské portály. Za povšimnutí stojí útoky z roku 2000, kdy některé společnosti jako Yahoo, Amazon či Ebay čelily útokům typu Distribued Denial of Service (DDoS) a nedávným cílem byly také servery vládních institucí. Během těchto útoků byly webové stránky jmenovaných společností nedostupné, což vedlo ke značným finančním ztrátám.

Útoky typu DoS se zaměřují predevším na aplikační nebo transportní vrstvu síťové architektury. Ve své práci se zabývám právě transportní vrstvou a typy DoS útoků, které jsou s ní spojené.

V první kapitole se zaměřím na teoretickou část, která obsahuje základní informace týkající se počítačových sítí. Popisuji zde počítačovou síť, síťovou architekturu a konkrétní protokoly.

V následující kapitole se věnuji útokům Denial of Service. Přiblížím jejich původ, fungování a vybané typy útoků. V druhé části kapitoly popíši strategie, pomocí kterých se lze před útoky Denial of Service bránit a jaké jsou cíle této obrany.

Ve třetí kapitole čtenáře seznámím s návrhem implementace aplikace, která je předmětem této bakalářské práce. Popisuji zde, jak jsem postupoval při návrhu a jakou by výsledná aplikace měla mít podobu.

V poslední a předposlední kapitole se zabývám samotnou implementací aplikace a následným testováním. Následuje shrnutí dosažených výsledků a naznačení směru, kterým by se mohla práce v budoucnu ubírat.

Kapitola 2

Úvod do problematiky počítačových sítí

Abychom pochopili, jak útoky DoS fungují, musíme si nejprve uvést, jak probíhá komunikace v počítačové síti. V této kapitole je rozebráno fungování počítačových sítí a síťová architektura. Konkrétně referenční model ISO/OSI, architektura TCP/IP a detailně jsou popsány jednotlivé vrstvy. Dále je rozebráno fungování protokolů TCP, UDP a ICMP, na které jsou útoky cíleny.

2.1 Počítačová síť

Počítačová síť, jak ji chápeme dnes, je zejména spojení mezi dvěma a více počítači, jenž mohou mezi sebou komunikovat a sdílet své prostředky. Tyto prostředky mohou být hardwarové či softwarové. Spojení je vytvořeno pomocí kabelu nebo bezdrátově. Komunikace mezi jednotlivými počítači se musí řídit podle určitých pravidel. V současné době, kdy jsou všechny sítě spojovány do globální celosvětové sítě Internet, je nejvíce využívána sada protokolů z rodiny TCP/IP.

Historie počítačových sítí sahá až do 60. let 20. století. V období Studené války se ministerstvo obrany USA snažilo vymyslet nový způsob komunikace, která by byla schopna ustát případný atomový útok a rovněž by zajišťovala komunikaci mezi vojenskými základnami tak, aby při případné destrukci jednoho z uzlů nebyla narušena komunikace celé sítě. Tato snaha roku 1969 vyústila ke vzniku sítě Arpanet [10], kterou zajistil úřad pro projekty pokročilého výzkumu ARPA. Síť Arpanet tedy poprvé fungovala na principu přepojování paketů.

Původním protokolem pro komunikaci v síti Arpanet byl Network Control Protocol (NCP), ale postupným vznikem pokročilejších technologií začal být nedostačující a postupně nahrazován propracovanějším standardem označovaným TCP/IP, který je používán dodnes.

Během 70. a 80. let získávaly přístup k výkonným počítačům mnohé sociální skupiny lidí. Standard TCP/IP byl volně k dostání a stále více lidí se připojovalo na rozšiřující se komplex sítí, který začal být znám pod označením Internet. Arpanet byl definitivně odpojen roku 1990.

2.2 Síťová architektura

Navrhnout a realizovat počítačovou síť je velice komplikovaná záležitost. Počítačová síť musí být kompatibilní, aby jednotlivé části sítě spolu mohly komunikovat, ale současně efektivní, aby byla dostatečně rychlá a nespotřebovávala přílis mnoho kapacity. Z tohoto důvodu byl zaveden takzvaný vrstvový model, který danou problematiku dekomponuje na dílčí hierarchicky uspořádané vrstvy, kde každá vrstva má na starost konrétní část dané problematiky. Na nejnižších vrstvách se řeší přenos jednotlivých bitů, zatímco na vyšších vrstvách pak přenos celých dat.

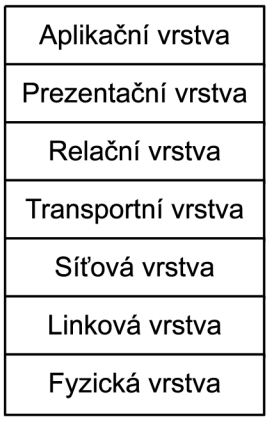
Zmíněný vrstvový model by nemohl fungovat, kdyby jednotlivé vrstvy mezi sebou neměly stanoveny pravidla komunikace. Každá vrstva v rámci daného uzlu může komunikovat pouze se svými bezprostředně sousedícími vrstvami. Každá z hierarchicky uspořádaných vrstev přidá data navíc k tomu, co poskytla vrstva bezprostředně nižší. Naopak v rámci jednotlivých uzlů v síti spolu mohou komunikovat pouze vrstvy na stejné úrovni. Pro vzájemnou komunikaci rovnocenných vrstev musí existovat pravidla komunikace. Tato pravidla se nazývají protokoly. Vlastností těchto protokolů je, že pracují na konkrétní vrstvě. Pro jednu vrstvu může být určeno i více protokolů, které nad ní pracují.

Je důležité, aby existovala všeobecně uznávaná dohoda o tom, kolik by mělo být hierarchicky uspořádaných vrstev a jaké role by měly vykonávat. Bylo vyvinuto několik síťových architektur, které odpovídaly tomuto vrstvovému modelu. Mezi nejznámější patří referenční model ISO/OSI, který trpěl mnoha nedostatky, a proto nebyl příliš úspěšný. Daleko populárnější byla architektura, která vycházela z rodiny protokolů TCP/IP a je na ní vybudován dnešní Internet.

Referenční model ISO/OSI

Open Systems Interconnection (OSI) referenční model [4] [12] byl navržen organizací International Organization for Standardization (ISO). Autoři referenčního modelu OSI předpokládali, že se tento model stane předpisem, ale to se nikdy nestalo, a tak přístup modelu Internetu a TCP/IP převládá.

Cílem referenčního modelu je poskytnout společnou základnu pro koordinované vypracovávání norem pro propojení systémů. Norma nestanovuje implementaci systémů, ale stanovuje všeobecné principy síťové architektury, která je rozdělena do sedmi vrstev. Každá vrstva vytváří rozhraní s nadřazenou a podřazenou vrstvou (dochází k výměně dat).



Obrázek 2.1: Referenční model ISO/OSI

Fyzická vrstva

Fyzická vrstva se stará o přenos jednotlivých bitů mezi příjemcem a odesílatelem pomocí komunikačního kanálu. Je tvořena vlastní kabeláží a jejími prvky. Kvalitní návrh a realizace je významným aspektem ovlivňující výkon celého komunikačního systému. Hlavním úkolem je vytvoření přenosové cesty mezi dvěma prvky sítě. V této vrstvě se jedná o definici signálů, které se používají k reprezentaci logické 1 a logické 0.

Linková vrstva

Linková vrstva zajišťuje spojení mezi dvěma sousedními prvky sítě. Vytváří rámce, které kromě vlastních přenášených informací obsahují údaje pro adresování, zabezpečení proti chybám přenosu a údaj o začátku rámce. Na této vrstvě pracují přepínače. Nejpoužívanějším protokolem linkové vrstvy je Ethernet.

Síťová vrstva

Síťová vrstva má na starost směrování dat v síti mezi zařízeními, které nejsou přímo propojené. Obsahuje funkce, které umožní překonat rozdílné vlastnosti technologií v přenosových sítích. Zajišťuje tedy volbu vhodné cesty přes mezilehlé uzly a předávání paketů po této trase až k příjemci. Nejznámějším protokolem síťové vrstvy je Internet Protocol (IP).

Transportní vrstva

Transportní vrstva zajišťuje samotný přenos dat k příslušnému aplikačnímu procesu na hostitelském počítači. Přijímá data z relační vrstvy, která následně rozkládá na pakety a následně je předává síťové vrstvě. Dále zabezpečuje, aby zpráva dorazila k příjemci celá. Zajišťuje i případné opakování zprávy v případě chyby. Mezi nejpoužívanější protokoly transportní vrstvy patří Transmission Pontrol Protocol (TCP) a User Datagram Protocol (UDP).

Relační vrstva

Relační vrstva navazuje a ukončuje relace mezi koncovými stanicemi. V rámci navázání relace si tato vrstva vyžádá vytvoření spojení od transportní vrstvy. Prostřednictvím tohoto spojení probíhá komunikace mezi účastníky relace. Tato vrstva se rovněž stará o řízení komunikace, včetně ukončení relace a zrušení spojení.

Prezentační vrstva

Data přenášená po síti mohou mít různou podobu, a protože jednotlivé počítače mohou používat rozdílnou reprezentaci těchto dat, je potřebná konverze přenášených dat. Prezentační vrstva se stará o komprimaci a šifrování dat. Úkolem této vrstvy je, aby přenášená data dorazila ke koncové stanici v takové podobě, v jaké byla vyslána.

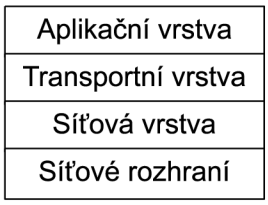
Aplikační vrstva

Tato vrstva je uživateli nejblíže. Narozdíl od ostatních vrstev nezajišťuje služby pro vyšší vrstvu, protože žádnou nemá. Funguje jako brána mezi aplikacemi běžících v různých uzlech, které si vzájemně vyměňují data. Úkolem této vrstvy je zajistit komunikaci mezi aplikacemi a umožnit tak jejich spolupráci.

Architektura TCP/IP

Rodina protokolů TCP/IP [17] vychází z referenčního modelu ISO/OSI, který jsem popsal v podkapitole 2.2. Narozdíl od modelu ISO/OSI je síťová komunikace rozdělena pouze do čtyř vrstev. Původně byl tento model určen primárně pro UNIX. Dnes je součástí prakticky všech operačních systémů a slouží ke komunikaci v síti Internet.

Tento model je nezávislý na přenosovém médiu, a proto je určen jak pro sítě Local Area Network (LAN), tak i Wide Area Network (WAN). Předpokládá, že na nižších vrstvách jsou přenosové služby nespolehlivé. Spolehlivost, pokud je vyžadována, zajišťují vrstvy vyšší. Obsahuje sadu protokolů, které jsou využity pro komunikaci v počítačové síti.



Obrázek 2.2: Architektura TCP/IP

Síťové rozhraní

Nejnižší vrstva síťové architektury TCP/IP. Tato vrstva zajišťuje přístup k přenosovému médiu a řízení datového spoje. Implementace je závislá na typu přenosové technologie, a proto TCP/IP tuto vrstvu blíže nespecifikuje. Předpokládá, že použitá řešení budou mimo rámec TCP/IP. Podle referenčního modelu ISO/OSI zahrnuje vrstvu fyzickou a linkovou.

Síťová vrstva

Síťová vrstva zajišťuje síťovou adresaci, směrování a předávání paketů. Hlavním požadavkem je především rychlost přenosu dat. Z tohoto důvodu síťová vrstva nezaručuje spolehlivé doručení přenášených dat. Na této vrstvě pracuje protokol IP, který je nespolehlivý a nespojovaný.

Transportní vrstva

Tato vrstva odpovídá transportní vrstvě referenčního modelu ISO/OSI. Úlohou této vrstvy je zajistit přenos dat mezi stanicemi. Mezi protokoly, které jsou využívány na této vrstvě, patří TCP a UDP.

Aplikační vrstva

Na této vrstvě probíhá provoz základních aplikací v rámci TCP/IP. Aplikační vrstva je nejvyšší vrstvou architektury TCP/IP a obsahuje všechny protokoly, které poskytují uživatelům konkrétní aplikace. Podle referenčního modelu ISO/OSI zahrnuje vrstvu aplikační, prezentační a relační.

2.3 Síťové protokoly

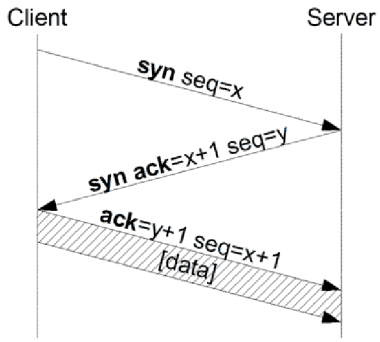
Síťové protokoly [17] představují sadu pravidel, podle kterých síť pracuje. Každý protokol má svůj vlastní způsob, jak jsou data formátována při odeslání a jak jsou tato data následně zpracována při přijetí. Existuje mnoho různých typů síťových protokolů. Každý pracuje na rozdílné vrstvě referenčního modelu ISO/OSI. Aby spolu mohly počítače v síti komunikovat, musí používat společný protokol.

Tato kapitola je zaměřena na protokoly TCP, UDP a ICMP, protože dále popsané útoky DoS budou cíleny právě proti těmto protokolům.

Transmission Control Protocol

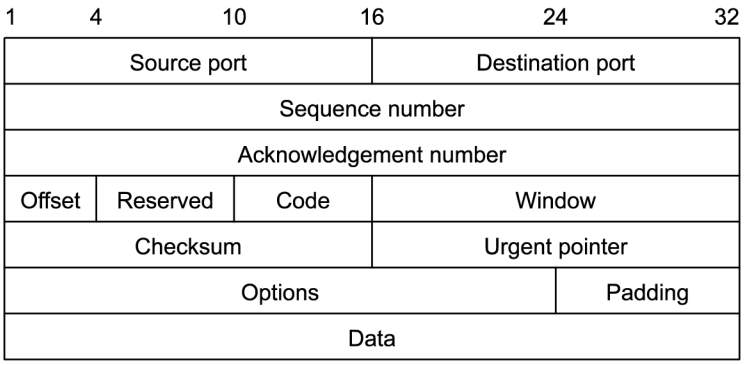
Transmission Control Protocol (TCP) [16] [7] je nejpoužívanějším protokolem transportní vrstvy, který zajišťuje spojovaný a spolehlivý obousměrný přenos dat nad nespolehlivými přenosovými cestami, kde existuje riziko poškození, ztráty, duplikace a nebo přeuspořádání přenášených paketů. Protokol garantuje spolehlivé doručování ve správném pořadí. TCP je využíván řadou aplikačních protokolů. Na obrázku 2.4 je znázorněna hlavička protokolu TCP.

Protože TCP je spojovaná služba, před zahájením odesílání dat musí být navázáno spojení mezi klientem a serverem. Toto spojení je navázáno pomocí takzvaného three-way handshake (Obrázek 2.3). To znamená, že navázání spojení probíhá ve tčech krocích. Klient zahájí komunikaci zasláním paketu s příznakem SYN, který obsahuje pořadové číslo prvního bajtu, které bude po navázání spojení odesílat. Pokud server přijme požadavek klienta, odešle paket s příznaky SYN a ACK. Na to klient odpoví paketem s příznakem ACK a spojení je úspešně navázáno.



Obrázek 2.3: Three-way handshake [3]

Uzavření spojení musí být rovněz potvrzeno z obou stran a probíhá podobným způsobem jako navázání spojení. Používá se k tomu příznak FIN a ACK. Strana, která vyžaduje ukončení spojení musí odeslat paket s nastaveným příznakem FIN. Strana, která obdržela paket FIN, odpoví datagramem s nastaveným příznakem ACK. Následně odešle další paket s nastaveným příznakem FIN, na který strana vyžadující spojení odopoví s nastaveným příznakem ACK. Po těchto čtyřech krocích je spojení považováno za uzavřené.



Obrázek 2.4: TCP hlavička

Source port

Toto 16–ti bitové číslo udává číslo portu, na kterém je spuštěna aplikace, která odeslala datagram.

Destination port

Toto 16–ti bitové číslo udává číslo portu, na kterém je spuštěna aplikace, která je příjemcem daného datagramu.

Sequence number

Toto 32–bitové číslo udává sekvenční číslo prvního data bajtu. Výjimku tvoří případ, kdy je nastaven kontrolní bit SYN. Pokud je tento bit nastaven v logiké 1, potom se jedná o počáteční sekvenční číslo a první data bajt je roven počátečnímu sekvenčnímu číslu + 1.

Acknowledgement number

Pokud je nastaven kontrolní bit ACK, potom 32–bitová hodnota tohoto pole udává sekvenční číslo následujícího datagramu. Jakmile je navázáno spojení, potom Acknowledgement number je vždy odeslán.

Offset

Počet 32–bitových slov v TCP hlavičce. Tato položka indikuje, kde v segmentu začínají data přenášená tímto datagramem.

Code

TCP používá šest kontrolních příznaků, kde každý příznak odpovídá jednomu bitu v tomto šesti bitovém poli. Příznaky jsou následující: URG, ACK, PSH, RST, SYN, FIN.

Window

Každý TCP datagram obsahuje tuto 16–ti bitovou hodnotu, která určuje maximální velikost přijatých dat jednoho datagramu v bajtech.

Checksum

Kontrolní součet, který slouží k ověření validity dat.

Urgent pointer

Toto pole určuje hodnotu ukazatele na urgentní data jako pozitivní offset od sekvenčního čísla v daném segmentu. Tato položka je interpretována pouze v případě, kdy je nastaven kontrolní příznak URG.

Options

Toto pole má proměnnou délku a obsahuje volitelné parametry. Využíváno je například pro indikaci maximální velikosti segmentu, kterou je přijímající strana schopna pojmout.

Padding

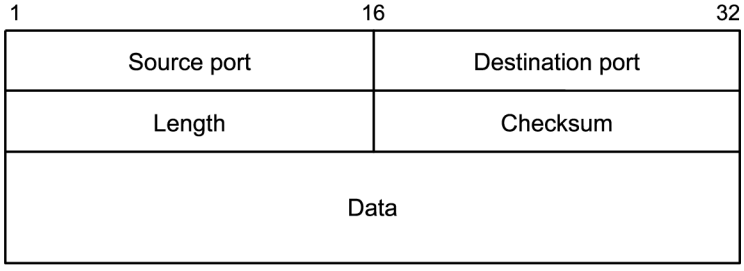
Obsahuje specifické množství nulových bitů, které doplňují hlavičku protokolu tak, aby byla beze zbytku dělitelná 32 a měla tak 32–bitovou hranici.

User Datagram Protokol

Nyní máme představu o tom, jak funguje protokol TCP, a tak se můžeme zaměřit na protokol UDP [14] [7] a porovnat rozdíly. Struktura hlavičky (Obrázek 2.5) je narozdíl od protokolu TCP odlišná. Účel jednotlivých polí v hlavičce bude dále vysvětlen.

Protokol UDP je stejně jako protokol TCP protokolem transportní vrstvy. Slouží stejně jako protokol TCP ke komunikaci dvou aplikací a používá síťový protokol IP pro spojení dvou zařízení. Obecně se dá říci, že protokol UDP poskytuje velmi jednoduché rozhraní mezi siťovou a aplikační vrstvou bez záruky doručení a bez ochrany proti duplikaci UDP datagramů. Protokol UDP identifikuje aplikace na zařízeních pomocí UDP portu, u kterého díky své unikátnosti můžeme jednoznačně rozpoznat a označit o kterou aplikaci se jedná.

Tento protokol, narozdíl od protokolu TCP, nemá žádnou možnost ovládání toku dat. Nejsou zde žádné segmenty a každý paket se jeví jako samostatná entita. To znamená, že způsob odesílání dat je závislý na aplikaci, která tato data odesílá. Vzhledem k povaze tohoto protokolu zde neexistuje žádné navazování spojení, a proto je označován jako protokol nespolehlivý. Data jsou odesílána přímo bez navázání spojení. Další chybějící funkcionalitou je možnost přeposlání. Protokol nemá žádnou možnost detekce, zda daný paket byl doručen.



Obrázek 2.5: UDP hlavička

Source port

Nepovinná 16–ti bitová položka, která při vyplnění indikuje port vysílacího procesu a zároveň zastupuje adresu, na kterou by měla být adresována odpověď v případě nedostupnosti dalších informací. Při nevyužití je vložena nulová hodnota.

Destination port

Tato 16–ti bitová položka slouží jako cílová adresa, která identifikuje proces na cílovém zařízení.

Length

Toto 16–ti bitové pole zastupuje délku datagramu v bajtech včetně hlavičky a dat (minimální velikost je 8 bajtů).

Checksum

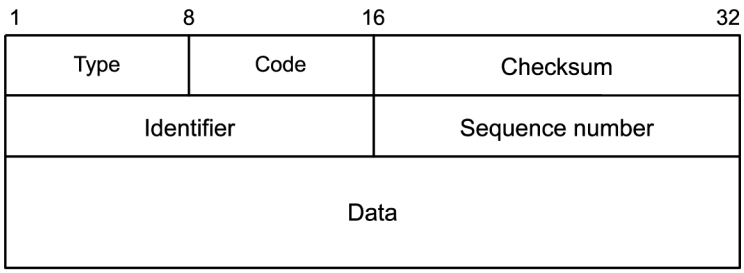
Kontrolní součet, který slouží k ověření, zda data nebyla poškozena.

Internet Control Message Protocol

Internet Control Message Protocol (ICMP) [15] [7] je jedním z nejdůležitějších internetových protokolů. Využívá základní podpory protokolu IP stejně, jako by to byl protokol vyšší vrstvy. Nicméne protokol ICMP je nedílnou součástí protokolu IP a musí být implementován každým modulem využívající protokol IP. Datagramy ICMP jsou přenášeny pomocí IP paketů – ICMP datagram je vkládán do IP paketu. Protokol IP pohlíží na ICMP datagram jako na data, která přenáší. Stejným způsobem IP protokol pohlíží třeba na UDP datagram nebo na TCP segment.

Protokol IP není navrhnut k tomu, aby byl spolehlivý. Z tohoto důvodu existuje protokol ICMP, aby poskytnul zpětnou vazbu při problémech v komunikaci. Zprávy ICMP jsou odeslány pouze v konkrétních případech. Například pokud datagram nemůže dorazit do cílové stanice, pokud výchozí brána nemá dostatečnou kapacitu paměti, aby mohla datagram dále přeposlat, a nebo pokud výchozí brána může nasměrovat uzel, aby data posílal po kratší trase.

Protokol ICMP typicky slouží k ohlášení chyb, které nastaly při zpracování datagramů. Pokud nastala chyba při odesílání ICMP zprávy, žádná další zprává ICMP již není odeslána, aby se předešlo nekonečné smyčce ICMP zpráv. Na obrázku 2.6 je znázorněna hlavička zprávy *Echo/Echo reply*, která se používá ke zjištění dosažitelnosti cílového zařízení. Tato zpráva je využívána při útocích DoS.



Obrázek 2.6: ICMP Echo/Echo reply hlavička

Type

Jedná se o 8–mi bitové pole jednoznačně udávající typ zprávy. Mezi typy zpráv patří například *Echo Reply*, *Echo Request* nebo *Destination Unreachable*.

Code

Tato 8–mi bitová hodnota specifikuje kód, který blíže určuje typ paketu a rozděluje zprávu na další podtypy, které zprávu blíže specifikují.

Checksum

Kontrolní součet zajišťující kontrolu validity dat.

Identifier

16–ti bitový identifikátor, pomocí kterého lze identifikovat danou zprávu. Hodnota může být nulová.

Sequence number

Pořadové číslo, pomocí kterého lze identifikovat danou zprávu. Toto 16 bitů dlouhé číslo plní stejnou funkci jako identifikátor.

Kapitola 3

Útoky DoS a ochrana

Protože jsme se již seznámili s problematikou a fungováním počítačových sítí, tak nyní si můžeme přiblížit samotné útoky DoS [1] [5] [6]. V této kapitole se čtenář seznámí s historií, vybranými útoky DoS a některými typy ochrany, které dokážou tyto útoky potlačit.

3.1 Původ útoků DoS

Útoky DoS nejsou pouhým slabým místem v síti Internetu, které by mohlo být opraveno malou změnou protokolu nebo sofistikovanou obranou na potenciálních cílových systémech.

Původ útoků DoS leží hluboko v samém jádru architektury Internetu, který byl navrhnut k tomu, aby byl funkční, nikoliv bezpečný. A v tomto ohledu plní svoji roli velmi dobře. Nabízí uživatelům rychlý, jednoduchý a levný komunikační prostředek na síťové úrovni, který nabízí protokolům službu „best effort“. Tato služba se označuje jako nezaručená, tedy nesplňující podmínky pro spolehlivou službu. Jediné, co můžeme spolehlivě říci je, že se pakety od odesílatele k příjemci pokusí přenést s maximálním možným úsilím. Ztráta, změna pořadí nebo poškození paketů jsou řešeny transportními protokoly umístěnými u odesílatele a příjemce, takzvaný „end-to-end“ princip.

Tyto dvě paradigma jsou stavebními kameny, na kterých byl Internet postaven. Problém nastává v případě, kdy jedna strana z pohledu „end-to-end“ principu se začne chovat tak, aby způsobila druhé straně škodu. V takovém případě jsou „end-to-end“ protokoly zneužity a přestanou plnit svoji funkci. Současně tak toto paradigma brání mezilehlé síti v tom, aby se zapojila a řídila provoz narušitele. Místo toho dále přeposílá pakety k jejich cílovým destinacím, kde přetíží zdroje oběti.

Tento problém začal být zřejmý v říjnu roku 1986, kdy Internet čelil sérii kolapsů ze zahlcení [13]. Přestože se problém začal rychle řešit návrhem a nasazením několika protokolů TCP se správou pro řízení zahlcování [8], koncová správa toků nebyla schopna zajistit spravedlivé rozdělení zdrojů. Tento problém byl zaznamenán a řešen pomocí směrovačů, které monitorovaly provoz a spravedlivě přidělovaly přenosové pásmo jednotlivým tokům.

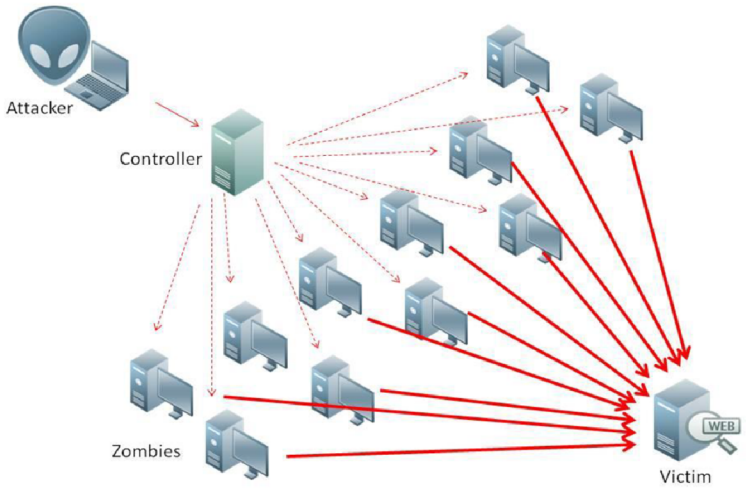
3.2 Útoky Denial of Service

Útok Denial of Service (DoS) je charakterizován jako pokus útočníka znepřístupnit konkrétní službu legitimním uživatelům. Existuje velká řada útoků DoS, což činí obranu velmi obtížnou. V podstatě každý počítač se může stát obětí útoku DoS. Útoky příchází v různých podobách a cílí na široké spektrum služeb. Tyto útoky by se daly rozdělit do tří karegorií:

- Spotřebování omezených nebo neobnovitelných zdrojů.
- Poškození nebo přenastavení konfiguračních informací.
- Fyzické poškození nebo přenastavení síťových komponent.

Tato práce je zaměřena na první typ útoku, tedy útoky, jenž spotřebovávají omezené nebo neobnovitelné zdroje. Cílem takových útoků může být přenosové pásmo, procesor, paměť, diskový prostor nebo kombinace těchto cílů. Tyto útoky mohou být klasifikovány jako útoky, které využívají zranitelnosti daného systému, a nebo jako útoky, jenž používají hrubou sílu. Tyto útoky nazýváme záplavovými.

Útoky zaměřující se na zranitelnost systému využívají jednu či více chyb ve vlastnostech nebo mechanismech, které zajišťují vlastnosti daného systému. Jejich cílem je především spotřeba omezených zdrojů, kterými daný systém disponuje. Toho je dosáhnuto odesláním vyrobených paketů, jenž dokážou těchto chyb využít. Naopak útoky používající hrubou sílu se snaží odepřít službu legitimnímu uživateli zaplavením cílového systému velkým množstvím paketů. Takové pakety se jeví jako validní a jejich cílem je zahltit přenosové pásmo.

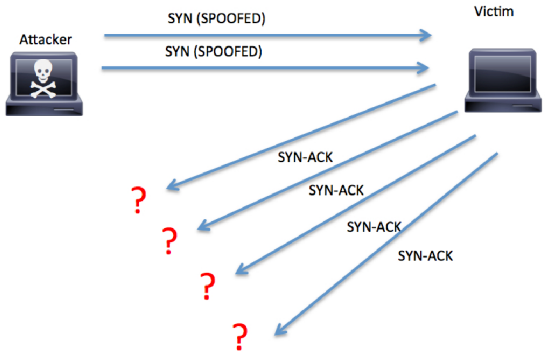


Obrázek 3.1: Distribued Denial of Service [2]

Na obrázku 3.1 je znázorněn útok Distributed Denial of Service (DDoS). Narozdíl od klasických útoků DoS, u kterých je útok generován z jednoho zařízení, jsou tyto útoky generovány z velkého množství stanic. Tyto stanice jsou rozprostřeny napříč celou sítí a obvykle jsou ovládány útočníkem z jednoho místa. Obecně je útoky DDoS mnohem obtížnější odrazit, protože přichází z velkého množství zařízení a páchají mnohem větší škody. Zahlcují přenosové pásmo, procesor a paměť oběti útoku.

SYN flood

Jedná se o jeden z nejrozšířenejších útoků. Využívá slabin v implementaci protokolu TCP. Používá se za účelem odstavení serveru a tím znemožnění zpracování požadavků na spojení od legitimních uživatelů. Jakákoliv služba využívající protokol TCP je zranitelná vůči útoku SYN flood.



Obrázek 3.2: Útok Syn flood [1]

Při normálním navázání spojení klient odešle serveru paket SYN. Následně server odpoví a odešle paket ACK a další požadavek SYN. Klient následně potvrdí spojení a odešle paket ACK. Po tomto procesu je spojení navázáno a klient může komunikovat se serverem. Jestliže klient neodpoví poslední zprávou ACK, server odešle znovu paket SYN ACK a čeká dokud klient neodpoví a nebo klientovi nevyprší čas. Toto spojení se nazývá napůl otevřené. A právě tohoto mechanismu je využíváno při útocích typu SYN flood.

Při útoku, který míří na tento prokotol se posílá pouze paket SYN. Server odpoví a zašle paket SYN ACK, nicméně už nikdy neobdrží finální paket ACK. Často je pozměněna zdrojová IP adresa, a tak server odpoví na nesprávnou adresu. Pokud obdržel paket SYN ACK počítač, který nevyžadoval spojení TCP, tak paket zahodí. Čím více takovýchto napůl otevřených spojení musí server obsloužit, tím méně mu zbývá prostředků. Server se stane nedostupný v případě, že mu zcela dojdou prostředky a nebude schopen otevřít nové spojení pro legitimní uživatele.

SYN ACK flood

Jedná se o další útok, ke kterému je využíván protokol TCP. K výměně dat pomocí protokolu TCP je vyžadováno navázané spojení. Jestliže spojení není navzáno, server příchozí pakety (kromě paketu SYN, který slouží k navázání spojení) zahodí. V případě odeslaní paketu SYN ACK, server odpoví a odešle paket RST, což značí, že spojení není navázáno. Tento útok spočívá v zasílání velkého množství takovýchto paketů s podvrženou IP adresou. Pokud se pakety dostanou až k serveru, server musí odpovědět zprávou RST, protože není navázáno spojení se serverem. Zpracování každého paketu stojí server určité prostředky. SYN ACK flood může být velice efektivní, stejně jako ostatní útoky na protokol TCP. Vede k zahlcení serveru, protože musí zpracovat každý paket.

UDP flood

UDP flood využívá k útoku prokol UDP. Útok typu UDP flood je jedním z nejjednodušších typů útoků. Cílem tohoto útoku je přetížení serveru nebo linky.

Princip tohoto útoku spočívá v tom, že útočník posílá na systém UDP pakety s náhodnými porty. Tyto pakety mívají většinou nestandardní velikost. Formát paketu není protokolem UDP specifikován a toho útočníci často zneužívají. Obvykle jsou data UDP paketu vyplněna velkým množstvím náhodných dat. To má za následek mnohem větší efek-

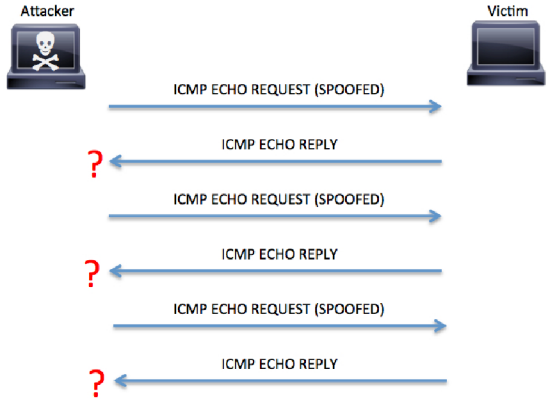
tivitu daného útoku. Systém, který je obětí útoku následně zpracuje přijaté pakety a snaží se na základě čísla UDP portu najít aplikaci, pro kterou je paket určen.

Pokud systém zjistí, že na požadovaném portu není spuštěna žádná příslušná aplikace, odpoví odesílateli zprávou ICMP o nedostupnosti *Destination Unreachable*. Cílem útoku je zahltit cílový systém UDP pakety, které nemají přiřazenu žádnou aplikaci. Na každý takový paket je následně odpovězeno zprávou o nedostupnosti. Z toho plyne, že čím více takových paketů je odesláno, tím efektivnější se útok stává, protože cílový systém je UDP pakety postupně zahlcen a tím i kapacita přenosového pásma.

Tento útok je v dnešní době velice populární a často využíván. Bývá zpravidla velmi efektivní a většinou stačí relativně malé zdrojové prostředky k jeho úspěšnému provedení.

ICMP flood

ICMP flood je jedním z nejstarších útoků DoS. Útok spočívá v zasílání masivního množství upravených *Echo request* zpráv s podvrhnutou zdrojovou IP adresou. Cíl následně odesílá odpovědi neexistujícím příjemcům a tím dochází k spotřebě systémových prostředků. Podvrhnutí zdrojové IP adresy není nutností, avšak odpovědi se budou koncentrovat zpět na útočníka, a proto je nezbytné, aby útočník disponoval větším výkonem než oběť. Další nevýhodou je možnost odhalení.

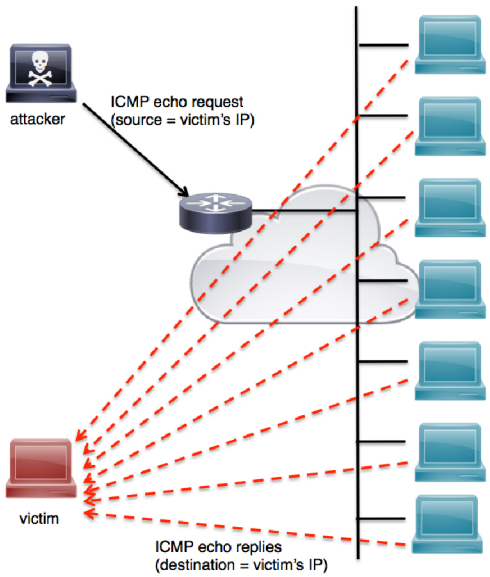


Obrázek 3.3: Útok ICMP flood [1]

Cílem útoku je zahlcení cílového systému a tím i zahlcení kapacity přenosového pásma. Útočník nečeká na odpovědi oběti a snaží se zprávy odesílat co nejrychleji. Efektivita útoku stoupá s počtem zařízení zapojených do útoku.

Útok Smurf

Jde o reflektivní zesilující útok. Princip je podobný jako ICMP flood, ale je zesílen. Tohoto zesílení je dosáhnuto tím, že požadavek *Echo request* je odeslán na IP adresu sítě s podvrhnutou zdrojovou IP adresou, která je pozměněna na IP adresu oběti. To má za následek, že všechny počítače z cílové sítě budou odpovědi směrovat k oběti útoku. Z toho vyplývá, že síla útoku je úměrná počtu zařízení připojených v dané síti.



Obrázek 3.4: Útok Smurf [1]

Tento útok již není v dnešní době příliš efektivní, protože používání adres sítí v Internetu je filtrováno, avšak stále existují výjimky, které tuto filtraci neprovádí.

3.3 Obrana před útoky DoS

Při střetu s útokem DoS je velmi důležité vědět, jak útok DoS funguje a jaké kroky provést při jeho prevenci a následné obraně. Tato kapitola se zabývá obranou proti útokům DoS a DDoS [11] [9].

Cíle obrany proti DoS/DDoS útokům

Tato část kapitoly je věnována několika cílům komplexních obranných řešení proti DoS a DDoS útokům.

Cíle obrany jsou následující:

- **Komplexnost** – Jedním z hlavních cílů kvalitní obrany proti síťovým útokům je komplexnost. Ideální ochrana by měla být schopná správně reagovat na všechny možné typy DoS/DDoS útoků, což je velmi těžké nebo prakticky nemožné splnit. Z hlediska efektivity je tedy tento cíl těžké splnit. Obranné řešení je většinou specializováno na danou metodu útoku a s jiným typem si neumí poradit. V konečném případě je tedy nutné zapojit více specifických řešení obrany.
- **Efektivita** – Další cíl je právě efektivita. Každá obrana musí být efektivní z hlediska poskytování dostatečné ochrany proti příchozímu útoku. Vzhledem k velkému množství metod útoku je i dosažení tohoto cíle velmi obtížné.
- **Přesnost odhadu provozu** – Obrana by měla být přesná v rozlišování mezi nevalidním a legitimním provozem. Pokud tento cíl obrana nesplňuje, může se stát, že v rámci

obrany přeruší veškeré služby legitimnímu uživateli. Obrana musí být schopná určit zda se jedná o útok a v nejideálnějším případě i o jaký typ útoku se jedná. Teprve v té chvíli může obrana začít s oddělením legitimního a nevalidního provozu.

- **Zajištění legitimního provozu** – Obrana musí být schopná zajistit dostupnost služeb oběti síťového útoku. Vzhledem ke komplexnosti útoků je tento cíl splněn většinou jen v případě, kdy je obrana postavena na specifický typ útoku. Pro ostatní typy útoku, než je tento specifický, není zajištění legitimního provozu v případě útoku možné splnit.
- **Obrana po ekonomické stránce** – V tomto cíli obrany jsou důležité ekonomické podmínky za jakých je daná obrana schopna efektivně chránit oběť útoku. Ekonomický cíl obrany je tedy takový cíl, kdy určité obranné řešení poskytuje dostatečnou obranu za odpovídající ekonomické náklady, vzhledem k následkům a výši škod potencionálního útoku.

Umístění obrany proti DoS/DDoS útokům

V této kapitole je popsáno umístění obrany v síti. Obrana se může nacházet na několika místech v síti. Provoz směřuje od systému útočníka přes směrovač do internetové sítě, kde pokračuje přes několik jiných routerů v internetu a následně přes firewall a jiné zařízení do routeru cílového zařizení a dále pak do cílového systému. Nejčastější umístění jsou – blízko zdroje útoku (útočníkova síť), blízko cíle útoku (sít oběti), mezi zdrojem a cílem a nebo hybridní umístění (spojení dvou nebo více předchozích umístění). Každé umístění má svoje výhody a nevýhody uvedené níže.

Umístění v síti útočníka

Umístění obrany v útočníkově síti znamená, že je obrana umístěna na hraničních směrovačích. Hraniční směrovače spojují síť poskytovatele připojení s internetovou sítí, kde se vyskytuje útočník. Vzhledem k efektivitě tohoto umístění obrany by bylo nejvhodnější, aby se obrana vyskytovala ve všech sítích, kde se vyskytuje útočník. Útočníci se většinou snaží rozmístit zdroje útoku do více sítí, aby zvýšili svoji šanci na úspěch a tím značně limitují efektivitu tohoto umístění.

Výhody:

- Malá agregace nevalidních paketů.
- Jednodušší rozlišitelnost validních a nevalidních paketů.

Nevýhody:

- Malá efektivita spojená s tím, že útočník může útočit z více sítí najednou.

Umístění v síti oběti

Obrana blízko cíle útoku je z pravidla umístěna v síti, kde se nachází cíl útoku. Jedná se o samotný systém oběti nebo o některý ze síťových prvků (firewall, směrovač, proxy server).

Výhody:

- Jednodušší možnosti detekce probíhajícího útoku. Obrana má v tomto umístění větší šanci na předání přesnějších informací.
- Snadné přizpůsobování obrany vzhledem k metodám útoku.

Nevýhody:

- Vysoká agregace nevalidních paketů, která může několikanásobně narůst oproti agregaci v útočníkově síti.
- Umístění obrany pokrývá pouze jeden potenciální cílový systém, který se může stát cílem útoku.
- Nutnost kvalifikovaného pracovníka, který musí provádět nastavení a modifikaci obranných prostředků.

Umístění mezi útočníkem a cílem

Obrana v rámci tohoto umístění bývá zapojena na routerech autonomních systémů. Hlavním cílem tohoto umístění je, aby byl útok detekován ještě dříve, něž se dostane k danému cíli.

Výhody:

- Široké pokrytí schopné chránit všechny prvky v dané cílové síti.

Nevýhody:

- Vysoké nároky na přesnost detekce, kde musí být obrana schopná správně rozlišit legitimní a nevalidní pakety. Při nesprávné funkci může tato obrana negaivně ovlivňovat více systémů v dané cílové síti a také v dalších okolních sítích.

Hybridní umístění obrany na více místech

Hybridní umístění obrany bylo zavedeno z dúvodu větší komplexnosti ochrany. Umístění na více místech kombinuje všechny tři předchozí zapojení a tvoří jeden velký obranný celek, kde mezi sebou jednotlivé obranné prvky komunikují. Toto umístění obrany je nejlepší volbou pro obecnou ochranu proti síťovým útokům. Obranný systém nejdříve detekuje probíhající síťový útok, následně identifikuje o jaký útok se jedná a začne se snažit o zajištění adekvátní ochrany co nejblíže u útočníka. Většina obranných řešení proti DoS a DDoS útokům pracuje právě na principu hybridního umístění.

Výhody:

- Komplexní obrana.
- Spolupráce jednotlivých prvků v obranné síti pro zajištění maximální ochrany v podobě jednoho obranného bloku.

Nevýhody:

- Vysoké nároky na správnou a přesnou koordinaci mezi jednotlivými prvky obranného bloku, což může mít za následek špatnou nebo žádnou funkčnost obrany.

Přístupy obrany proti útokům

V této části se budeme zabývat třemi hlavními fázemi obrany proti síťovým útokům DoS/DDoS.

Prevence

Prevence je často označována jako nejdůležitější fáze obrany proti DoS a DDoS útokům. Zahrnuje dva přístupy. Jedním přístupem je přijetí takových preventivních opatření, aby nebylo možné konkrétní útok provést. Druhým přístupem je snaha o zmenšení efektivity útoku na minimum. Hlavním cílem prevence je co nejefektivněji zamezit a předejít DoS a DDoS útokům. Tato opatření mohou být umístěna na jekémkoliv místě v síti.

- **Odstranění slabých míst na cílovém systému a v jeho síti** – Tento přístup usiluje o odstranění slabých míst, které mohou představovat potenciální cíl útoku. Konkrétně toto opatření může zahrnovat instalaci aktualizací softwaru, zapnutí firewallu, systému IPS (Intrusion Prevention System) apod.
- **Filtrování nelegitimních paketů** – V tomto obranném přístupu dochází k zapojení zařízení, které provádí detailní inspekci přijímaných a odesílaných paketů. Pakety, které nesplňují určité požadavky, jsou zahozeny.
- **Source Validation** – Ověření identity zdroje je dalším preventivním obranným přístupem. Tento obranný přístup se snaží rozpoznat techniku podvržení zdrojových IP adres a tím rozeznat, jestli se jedná o člověka. Ověření bývá provedeno například pomocí výběru správného obrázku, vyřešení jednoduché rovnice, zapsáním jednoduchého textu apod.
- **Resource Allocation** – V tomto obranném přístupu server všem klientům, kterým poskytuje služby, přiřadí stejný objem zdrojových prostředků a tím se snaží předejít potenciálnímu útoku. V rámci tohoto obranného přístupu je často využíváno i ověření identity, kvůli tomu, že se útočník často snaží falšovat zdrojové IP adresy a pro server se tváří jako další legitimní uživatel a připravuje tím server o zdroje.
- **Overprovisioning** – Tento obranný přístup se snaží připravit cílové systémy a síťové prvky na několikanásobně větší zatížení, než které odpovídá obvyklému provozu. Cílem toho přístupu je případný útok přečkat bez negativních efektů na systém.

Detekce

Detekce je druhou hlavní fází obrany proti DoS/DDoS útokům. Jedná se o fázi, která probíhá současně s probíhajícím útokem. Nejlepším umístěním přístupu obrany pro detekci je u cíle útoku, kde dokáže poskytnout nejpřesnější informace o dané metodě útoku. Detekce je rozdělena na dva kroky. Nejdříve jde o rozpoznání toho, že jde o DoS/DDoS útok a následně o co nepřesnější charakteristiku metody daného útoku. Detekce má dva hlavní cíle kterými jsou přesnost a rychlost.

Rozlišujeme následující přístupy detekce:

- **Signature Detection** – Obranný přístup využívá podle předem určitých pravidel vytvořenou databázi, ve které jsou uloženy záznamy o známých DoS/DDoS útocích.

K vytvoření databáze jsou využívány informace o proběhlých útocích. Každý záznam je pak porovnáván s pakety procházející detekcí a při nalezené shodě obranný přístup detekuje, že se jedná o DoS/DDoS útok.

- **Anomaly Detection** – Principem tohoto obranného přístupu je vytvoření modelu legitimního síťového provozu, který je porovnáván s reálným síťovým provozem a při nalezení neshody detekuje tento obranný přístup probíhající útok. Nevýhodou je, že při využívání nesprávného modelu může dojít k situacím, kdy většina nevalidních paketů není rozpoznána a nebo naopak jsou legitimní pakety označeny jako nevalidní.

Reakce

Třetí a poslední fází obrany proti síťovým útokům typu DoS a DDoS je reakce. Reakce přichází na řadu po fázi detekce. Cílem této fáze je adekvátně odpovědět na probíhající útok tak, aby došlo k zmírnění nebo k úplnému zastavení útoku. Většina obranných přístupů nedokáže probíhající útok zcela zastavit a soustředí se tedy na maximální možné zmírnění průběhu útoku a jeho případných následků na chod a přístup k cílovému systému. Obranné přístupy zahrnují dva kroky. Prvním principem je identifikace zdroje útoku a druhým odpovídající reakce na daný typ útoku.

Hlavní přístupy reakce:

- **Traceback** – Traceback má za úkol identifikovat zdroj síťového útoku. Na základě této identifikace pak dojde k adekvátní reakci v podobě dvou přístupů popsaných níže.
- **Rate limiting** – Zavedení omezení pro objem přijímaných dat odesílaných paketů, kde jsou omezeny všechny pakety, které byly detekcí označeny jako nevalidní. Tento princip se používá, když si nejsme jisti přesností výsledků detekce a je lepší dané pakety pouze omezit než riskovat úplným odmítnutím omezení služeb legitimního uživatele.
- **Filtrování** – Tento obranný přístup na rozdíl od Rate limiting všechny označené pakety rovnou odfiltruje a zahazuje. Označený paket je okamžitě ztracen a systém se s ním už dále nezabývá. Filtrování je využíváno v případě, že detekce poskytne přesné výsledky a můžeme si být jisti, že nevalidní pakety jsou správně označeny.

Kapitola 4

Návrh implementace

Obsahem této kapitoly je analýza a následný návrh řešení aplikace, která je předmětem této práce. Po nastudování problematiky počítačových sítí a útoků DoS jsem mohl přistoupit k samotnému návrhu aplikace. Při návrhu jsem vycházel z již existujících nástrojů pro generování útoků typu DoS. Zejména nástroj `hping3`[1], který je určen pro systém Linux.

4.1 Analýza požadavků

Cílem je vytvořit aplikaci, která bude schopna vygenerovat útoky typu Denial of Service. Tato aplikace by měla využít podporu knihovny `Pcap`, kterou podrobně rozeberu v kapitole 4.4. Rozhodl jsem se implementovat útoky SYN flood, ICMP flood a UDP flood, protože patří v dnešní době mezi nejvíce rozšířené útoky. Aplikace bude schopna vygenerovat vybrané útoky a tím se pokusit vyřadit zařízení, které bude cílem takového útoku. Hlavní požadavky na aplikaci jsou následující:

- Generování útoků SYN flood, UDP flood a ICMP flood.
- Přenesitelnost zdrojového kódu mezi platformami Windows a Linux.
- Efektivní využívání systémových prostředků a rychlost aplikace.

Dále by aplikace měla umožňovat zvolení zdrojové IP adresy. Buď jako samotnou adresu nebo rozsah adres, ze kterých bude náhodně generována. Pokud nebude specifikována, tak bude adresa zcela náhodná. Vstupní parametry bude rovněž možné zadat pomocí vstupního XML souboru, kde cesta k tomuto souboru bude speficikována parametrem příkazové řádky.

4.2 Struktura aplikace

Aplikace bude rozčleněna do několika částí, které budou vzájemně spolupracovat. Při implementaci bude kladen velký důraz na správný objektový návrh, díky kterému mohu aplikaci navrhnout s velkou mírou abstrakce. Dále bude aplikace implementována s cílem znovupoužitelnsti. Zejména část, která se bude starat o skládání paketů, protože může být použita v dalších aplikacích.

Průchod aplikací bude rozdělen do několika částí:

[1] hping3 - http://www.hping.org/hping3.html

- Zpracování vstupních parametrů,
- Otevření adaptéru,
- Vytvoření paketu,
- Hlavní smyčka odesílání paketu.

4.3 Vstupní XML soubor

Kromě příkazové řádky bude aplikace přijímat konfiguraci útoků pomocí vstupního XML souboru. Cesta k tomuto XML souboru bude zadána pomocí parametru příkazové řádky. Formát je popsán následujícím úsekem kódu v jazyce Document Type Definition (DTD):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE dos [

<!ELEMENT dos (synflood | udpflood | icmpflood)>

<!ELEMENT synflood (source?, victim, payload?, source_port?, destination_port?)>
<!ELEMENT udpflood (source?, victim, payload?, source_port?, destination_port?)>
<!ELEMENT icmpflood (source?, victim, payload?)>

<!ELEMENT source (#PCDATA)>
<!ELEMENT victim (#PCDATA)>
<!ELEMENT payload (#PCDATA)>
<!ELEMENT source_port (#PCDATA)>
<!ELEMENT destination_port (#PCDATA)>
]>
```

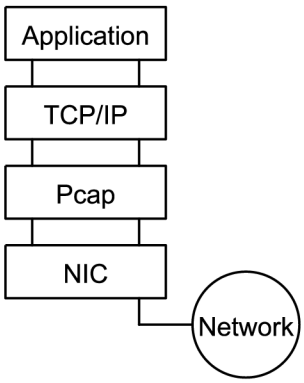
Hodnota elementů `source`, `source_port` nebo `destination_port` bude specifikována samostatně a nebo jako rozsah, kde hodnota bude náhodně generována ze specifikovaného rozsahu[2].

4.4 Knihovna Pcap

Knihovna Pcap[3] vytváří API pro čtení a zápis dat na síťové rozhraní. Pro systémy Unix je implementována knihovna libpcap. Pro systémy Windows je implementována knihovna WinPcap. Obě verze jsou implementovány v jazyce C. Pomocí této knihovny můžeme přistupovat ke všem paketům na síti. I k těm, které míří k jiné stanici na daném segmentu (čtení v promiskuitním režimu). Knihovna neobsahuje rutiny pro vytváření paketů TCP, IP datagramů či linkových rámců. To znamená, že jednotlivé pakety musíme vytvořit ručne. Právě proto se tato knihovna používá převážně pro odchytávání dat.

[2] Příklad rozsahu IP adres: `<source>192.168.1.1-192.168.1.254</source>`

[3] Tcpdump & Libpcap – http://www.tcpdump.org/



Obrázek 4.1: Pcap

Tato knihovna přistupuje přímo k síťové kartě, ze které čte data. Obrázek 4.1 znázorňuje na jaké vrstvě se nachází. Pro moji práci nejvýznamnější funkcí je `pcap_sendpacket`, která odešle ručně sestavený paket skrze síťové rozhraní. Další významnou funkcí je funkce `pcap_open_live`, která otevře síťové rozhraní pro odchytávání či vysílání paketů.

Kapitola 5

Implementace a testování

Před zahájením samotné implementace bylo zcela klíčové vybrat vhodný programovací jazyk. Mezi hlavní požadavky při výběru programovacího jazyka patřily rychlost, možnost efektivní práce s pakety na nižší úrovni ISO/OSI modelu a přenositelnost zdrojového kódu. Jako nejvhodnější se jevil jazyk C++11 díky existujícím knihovnám pro práci s pakety, a proto jsem se jej rozhodl pro implementaci použít.

Při implementaci jsem kladl velký důraz na přehlednost a čitelnost zdrojového kódu, na efektivní práci s pamětí a optimalizaci samotné aplikace. Dále na správný objektový návrh, vysokou míru abstrakce a znovupoužitelnost zdrojového kódu.

Celá aplikace je rozdělena do několika dílčích částí, které jsou na sobě závislé, avšak zcela oddělené. Jednotlivé části popíši v následujících kapitolách. Aplikace je závislá na knihovně `Pcap`, pomocí které otevře adaptér a vytvořené pakety odesílá.

5.1 Vstupní parametry

Zpracování vstupních parametrů je zajištěno volně dostupnou knihovnou `The Lean Mean C++ Option Parser`[1]. Tato knihovna je zcela samostatná a není závislá ani na standardní knihovně C++. Díky tomu je snadno použitelná pro operační systém Windows i Linux. Jedním ze vstupních parametrů je parametr `--file`, pomocí kterého lze specifikovat vstupní soubor ve formátu XML. Zpracování vstupního XML souboru je provedeno za pomoci jednoduché a efektivní knihovny `TinyXML-2`[2]. Tato knihovna byla do aplikace snadno integrovatelná. Po zpracování vstupních parametrů je naplněna datová struktura `Configuration`, která je následně předána hlavnímu programu ke zpracování.

Parametry příkazové řádky

Ovládání aplikace je implementováno pomocí následujících parametrů příkazové řádky:

- **--help** – Vypíše nápovědu na standardní výstup a ukončí aplikaci.
- **--adapters** – Vypíše seznam zařízení na standardní výstup a ukončí aplikaci.
- **--adapter**=*hodnota* – Povinný parametr, kde hodnota specifikuje index síťového zařízení, které bude použito.

[1] The Lean Mean C++ Option Parser – http://optionparser.sourceforge.net/
[2] TinyXML-2 – http://www.grinninglizard.com/tinyxml2/

- **--synflood, --udpflood, --icmpflood** – Specifikuje typ útoku. Aktivní může být pouze jeden z těchto povinných přepínačů.
- **--file**=*hodnota* – Specifikuje cestu ke konfiguračnímu XML souboru.
- **--srcip**=*hodnota* – Volitelný parametr, který určuje zdrojovou IP adresu, případně rozsah IP adres. Pokud parametr není přítomen, tak zdrojová ip adresa je generována náhodně.
- **--dstip**=*hodnota* – Povinný parametr specifikující cíl útoku.
- **--payload**=*hodnota* – Pomocí tohoto volitelného parametru lze určit velikost náhodně vygenerovaných dat, která budou odeslána současně s hlavičkou paketu.
- **--srcport**=*hodnota* – Volitelný parametr, který specifikuje zdrojový port nebo rozsah zdrojových portů.
- **--dstport**=*hodnota* – Volitelný parametr, který specifikuje cílový port nebo rozsa hcílových portů.

Pokud se jedná o rozsah hodnot, tak jsou tyto krajní hodnoty odděleny čárkou[3].

5.2 Výběr adaptéru

Pro zahájení útoku je nutné specifikovat adaptér, ze kterého bude útok generován. Zobrazení všech dostupných zařízení zajišťuje veřejná statická funkce `printDevices`, jenž náleží veřejné třídě `Adapter` a vyvolat ji lze zadáním volitelného parametru `--adapters`. Funkce projde dynamický lineární seznam a vypíše všechna dostupná zařízení, která poskytla funkce `pcap_findalldevs` z knihovny `Pcap` na standardní výstup a příslušné indexy, podle kterých se daný adaptér identifikuje. Poté provede ukončení aplikace. Příslušný index, kerý specifikuje vybraný adaptér je poté dosazen jako hodnota parametru `--adapter`.

5.3 Získání MAC adres

Tato část aplikace byla z programátorského hlediska kritická. Neexistuje jednotný způsob, jak adresu získat, protože tuto funkcionalitu si zajišťuje samotné jádro systému. Z toho důvodu se implementace pro systémy Windows a Linux liší. Možné řešení by bylo sestavit si vlastní `ARP` paket a pomocí knihovny `Pcap` jej odeslat a následně zachytit odpověď, ze které bych potřebnou adresu získal, ale vzhledem k existenci již hotových knihoven pro zmíněné operační systémy je toto řešení nevhodné.

Získání MAC adres zajišťuje instance třídy `Adapter`. Při vytvoření instance této třídy se zavolá privátní metoda `_getDevice`, jenž za pomoci funkce `getMacAddress`, která patří do jmenného prostoru `Multiplatform`, získá zdrojovou a cílovou MAC adresu na základě cílové IP adresy, která byla získána při zpracování vstupních parametrů. Tato funkce dále volá příslušné funkce z jmenného prostoru `WindowsApi` či `LinuxApi` na základě platformy, na které je zdrojový kód překládán.

Pro platformu Windows jsem využil knihovnu `iphlpapi.h`. Tato knihovna je součástí vývojářského balíku `Microsoft Platform SDK`, jenž poskytuje knihovny a nástroje určené k vývoji aplikací pro platformu `Windows`. Funkce `GetAdaptersInfo` vrátí informace

[3]Příklad rozsahu hodnot: --srcport=1200,2000

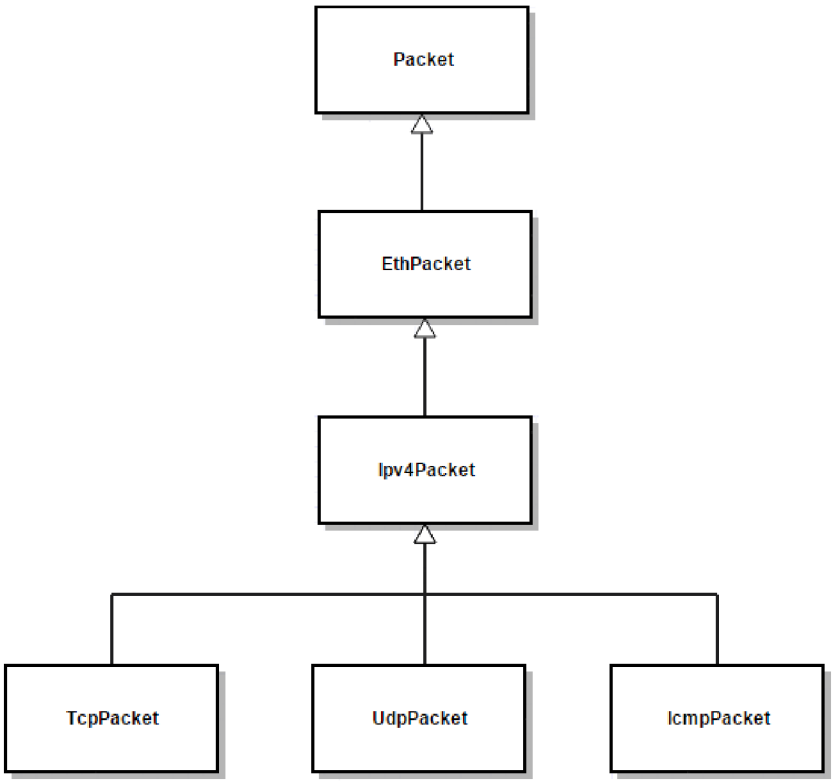
o všech dostupných adaptérech. Po nalezení příslušného adaptéru, který byl specifikován jako vstupní parametr získám zdrojovou MAC adresu. Pro získání cílové MAC adresy používám funkci `SendARP`, jenž odešle ARP paket na specifikovanou IP adresu a zjistí příslušnou MAC adresu. Pokud zadaná cílová IP adresa nepatří do stejné sítě jako adresa adaptéru, tak získám adresu výchozí brány.

Pro platformu Linux jsem se rozhodl použít knihovnu `linux/rtnetlink.h`, pomocí které odešlu požadavek ARP a díky funkci `ioctl` z knihovny `sys/ioctl.h` získám cílovou MAC adresu. Zdrojová MAC adresa je získána vytvořením socketu, kde jádro systému automaticky vyplní zdrojovou MAC adresu a pomocí funkce `ioctl` ji snadno získám.

5.4 Vytvoření paketu

Při vytváření paketu jsem se zaměřil na co nejefektivnější práci s pamětí, protože při záplavových útocích DOS je rychlost naprosto klíčová. Z toho důvodu pracuji pouze se statickou pamětí, protože je rychlejší než paměť dynamicky přidělená. Celou třídu, která se stará o vytváření paketů, jsem implementoval s cílem znovupoužitelnosti v dalších aplikacích.

Základ paketu je tvořen třídou `Packet`, jenž obsahuje pole reprezentující daný paket o velikosti 65536 bajtů, což je maximální velikost paketu. Z této třídy dále dědí další třídy reprezentující jednotlivé pakety. Dědičnost je znázorněna na obrázku 5.1.



Obrázek 5.1: Diagram tříd

Každá třída reprezentující daný paket obsahuje ukazatel na datovou strukturu, která

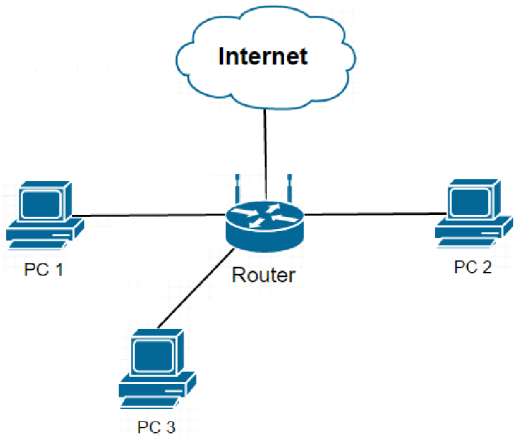
představuje hlavičku konkrétního protokolu. Při vytvoření instance takové třídy je tento ukazatel inicializován, aby ukazoval na konkrétní místo ve statickém poli ze třídy `Packet`. Pozice tohoto místa je spočítána jako součet velikostí struktur představující hlavičky protokolů, které jsou na nižší vrstvě v ISO/OSI modelu. Tím je docíleno uspořádání paketu podle stanovených konvencí. Paket dále obsahuje veřejné metody specifické pro konkrétní paket. Pomocí těchto metod lze nastavovat a číst atributy jednotlivých položek datové struktury představující hlavičku protokolu.

5.5 Implementace útoků

Základem pro všechny útoky je třída `Flood`. Tato třída obsahuje instanci třídy `Adapter` a strukturu, která je naplněna vstupními parametry. Z této třídy dále dědí `SynFlood`, `UdpFlood` a `IcmpFlood`. Každá z těchto tříd reprezentuje specifický útok. Implementace je velmi podobná. Prvně dojde k vytvoření instance dané třídy a inicializaci počátečních hodnot. Včetně vytvoření paketu a získání nezbytných hodnot z adaptéru. Po zavolání instanční metody `start` se zahájí samotný útok. Tento útok je prováděn v nekonečné smyčce, kde atributy příslušné struktury reprezentující paket se generují na základě vstupních parametrů. Aby aplikace byla co nejefektivnější, tak se pracuje s jednou instancí paketu, která se odesílá skrze síťový adaptér v nekonečné smyčce.

5.6 Testování

V této kapitole se budu důkladně zabývat testováním aplikace, kterou jsem navrhnul a implementoval. Testování probíhalo v uzavřené lokální wifi síti, protože útoky na veřejnou síť jsou zakázané. Na obrázku 5.2 je znázorněna topologie testovací sítě. Veškerá komunikace oběti útoku, která směřovala mimo lokální síť, byla směrovačem filtrována.



Obrázek 5.2: Topologie sítě

Testovací sestava

Významný vliv na rychlost generování útoků má sestava, ze které jsou útoky vysílány. Testování jsem prováděl na sestavě se čtvrtou generací procesoru Intel Core i5 se čtyřmi

fyzickými jádry a frekvencí 3,4 GHz, 2x 4GB DDR3 pamětí s rychlostí 1600 MHz a grafickou kartou GeForce GTX 760. Testování probíhalo v operačním systému Windows 10.

Tesování útoku SYN flood

Při generování tohoto útoku jsem zaplavil cíl TCP SYN pakety s podvrhnutou zdrojovou IP adresou. Monitorováním provozu v systému Linux jsem zjistil, že jádro přijaté SYN ACK pakety ruší zasláním paketu RST, a proto je nezbytné, aby zdrojová IP adresa byla podvrhnuta a paket se k zdánlivému odesílateli nikdy nedostal.

Zařízení v síti

- **PC 1** – Apache server naslouchající na portu 80, IP adresa – 192.168.0.11.
- **PC 2** – Útočník, IP adresa – 192.168.0.13.
- **PC 3** – Uživatel přistupující ke službě na PC 1, IP adresa – 192.168.0.16.

Konfigurace

Aplikaci jsem spustil na zařízení **PC 2**. Zadaná cílová IP adresa byla 192.168.0.11 a cílový port 80. Generování zdrojové IP adresy a portu bylo zcela náhodné.

Průběh útoku

Po zahájení útoku server rezervoval část svých prostředků pro každou příchozí žádost (Obrázek 5.4) o spojení. Na obrázku 5.3 je zachycena snaha serveru o potvrzení spojení. Po chvíli tato žádost byla zamítnuta odesláním paketu RST a následně došlo k uvolnění prostředků přidělených pro tuto žádost.

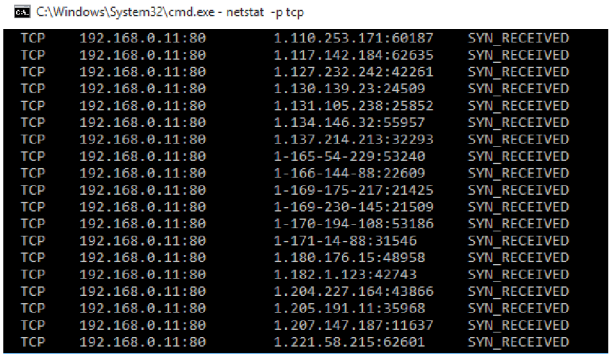
8	3.85914500	16.227.209.170	192.168.0.11	TCP	54 18923→80 [SYN] Seq=0 Win=512 Len=0
9	3.85934700	192.168.0.11	16.227.209.170	TCP	58 80→18923 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
27	6.86028400	192.168.0.11	16.227.209.170	TCP	58 [TCP Retransmission] 80→18923 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
36	12.8605580	192.168.0.11	16.227.209.170	TCP	58 [TCP Retransmission] 80→18923 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
63	24.8611810	192.168.0.11	16.227.209.170	TCP	54 80→18923 [RST] Seq=1 Win=0 Len=0

Obrázek 5.3: Reakce serveru na přijatý paket SYN.

129843	46.7988090	232.83.83.245	192.168.0.11	TCP	54 13828→80 [SYN] Seq=0 Win=512 Len=0
129844	46.8000880	131.39.235.138	192.168.0.11	TCP	54 12530→80 [SYN] Seq=0 Win=512 Len=0
129845	46.8009480	185.132.153.107	192.168.0.11	TCP	54 21664→80 [SYN] Seq=0 Win=512 Len=0
129846	46.8029500	164.4.107.126	192.168.0.11	TCP	54 35579→80 [SYN] Seq=0 Win=512 Len=0
129847	46.8048650	201.34.106.200	192.168.0.11	TCP	54 18700→80 [SYN] Seq=0 Win=512 Len=0
129848	46.8068960	243.0.48.19	192.168.0.11	TCP	54 31073→80 [SYN] Seq=0 Win=512 Len=0
129849	46.8090520	218.235.57.169	192.168.0.11	TCP	54 45412→80 [SYN] Seq=0 Win=512 Len=0
129850	46.8117890	255.25.218.193	192.168.0.11	TCP	54 54811→80 [SYN] Seq=0 Win=512 Len=0
129851	46.8124780	194.233.19.134	192.168.0.11	TCP	54 4978→80 [SYN] Seq=0 Win=512 Len=0

Obrázek 5.4: Příchozí pakety SYN.

To mělo za následek, že server uchovával informace o napůl otevřených spojeních (Obrázek 5.5) a postupně spotřebovával své cenné prostředky, protože žádosti o spojení přicházely rychleji než server stíhal neplatné žádosti rušit.



Obrázek 5.5: Napůl otevřená TCP spojení.

Server rychle spotřeboval všechny své prostředky a stal se nedostupný. Útok byl úspěšný a uživatel, který se snažil navázat spojení ze zařízení **PC 3** zůstal neobsloužen. Spojení se nepodařilo navázat.

Tesování útoku ICMP flood

Při generování tohoto útoku jsem zahltil cíl velkým množstvím upravených *Echo request* zpráv s podvrhnutou zdrojovou IP adresou. Pro zesílení útoku byla k hlavičce paketu přidána náhodně vygenerovaná data.

Zařízení v síti

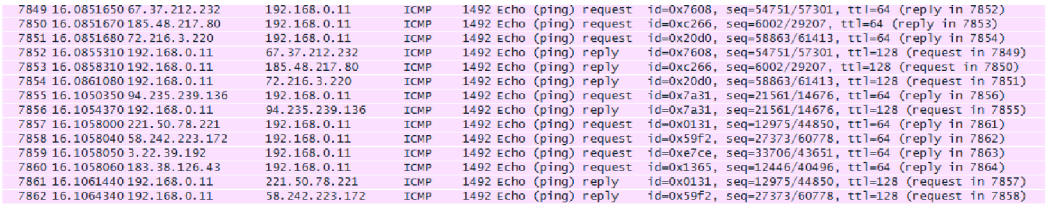
- **PC 1** – Uživatel, který se stal obětí útoku, IP adresa – 192.168.0.11.
- **PC 2** – Útočník, IP adresa – 192.168.0.13.
- **PC 3** – Uživatel snažící se kontaktovat zařízení PC 1, IP adresa – 192.168.0.16.

Konfigurace

Aplikaci jsem spustil na zařízení **PC 2**. Zadaná IP adresa oběti útoku byla 192.168.0.11. Pro maximální efektivitu útoku jsem nastavil velikost dat na 1400 bajtů. Generování zdrojové IP adresy bylo zcela náhodné.

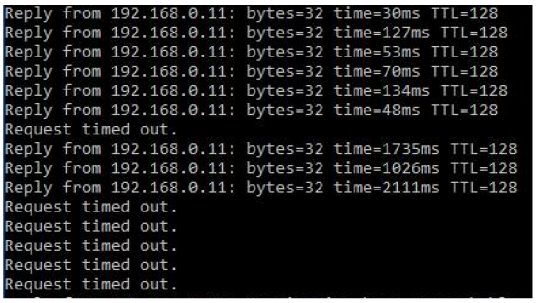
Průběh útoku

Útočník začal oběť zahlcovat ICMP zprávy *Echo request*, kde každá zpráva měla celkovou velikost 1492 bajtů. Počítač, jenž se stal obětí útoku na tyto podvrhnuté zprávy odpovídal (Obrázek 5.7) a tím byl útok zesílen, protože se přenosová cesta začala zahlcovat oboustranně.



Obrázek 5.6: Zprávy *Echo request* a odpovědi.

Při ověřování funkčnosti spojení mezi zařízením **PC 3** a **PC 1** došlo po zahájení útoku k velké prodlevě mezi odpověďmi. Poté byla přenosová cesta cílového zařízení zcela zahlcena a spojení ztraceno. (Obrázek 5.7). Rychlost útoku, který byl generován na straně útočníka dosahoval rychlosti až 200 Mb/s.



Obrázek 5.7: Ověření funkčnosti spojení.

Objem příchozích dat na straně oběti útoku způsobil znatelné navýšení zátěže procesoru. Po chvíli došlo k úplnému zkolabování systému a zařízení muselo být restartováno.

Tesování útoku UDP flood

V tomto testovacím případě útočící stanice cílila na multimediální datový proud, který byl vysílán pomocí protokolu UDP. Cílem bylo narušit datový proud a uživateli překazit sledování vysílaného video přenosu.

Zařízení v síti

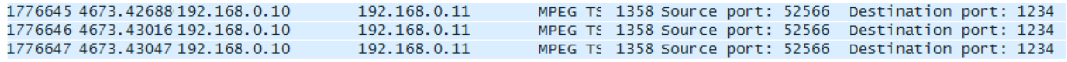
- **PC 1** – Uživatel, který se stal obětí útoku, IP – adresa 192.168.0.11.
- **PC 2** – Útočník, IP adresa – 192.168.0.13.
- **PC 3** – Stanice vysílající multimediální datový proud, IP adresa – 192.168.0.10.

Konfigurace

Aplikaci jsem spustil na zařízení **PC 2**. Zadaná IP adresa oběti útoku byla 192.168.0.11. Zdrojová IP adresa byla nastavena stejná jako zařízení **PC 3** (192.168.0.10). Dále cílový port byl nastaven na hodnotu 1234 a velikost generovaných dat na 500 bajtů.

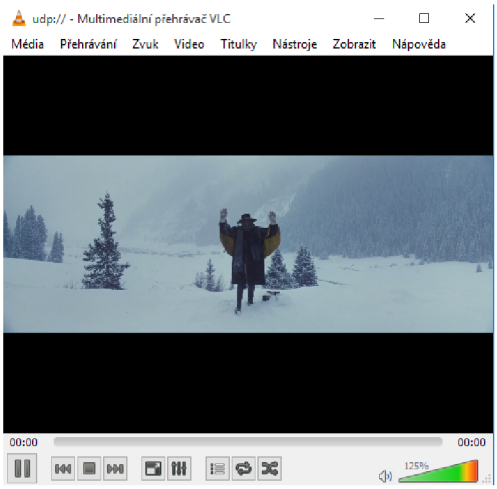
Průběh útoku

Stanice **PC 3** vysílala multimediální datový proud, který byl určen pro stanici **PC 1**. Validní pakety, které zachytila tato stanice, jsou vidět na obrázku 5.8.



Obrázek 5.8: Validní pakety multimediálního datového proudu.

Na Obrázku 5.9 je náhled na video přenos, který dosud nebyl napaden útokem UDP flood.



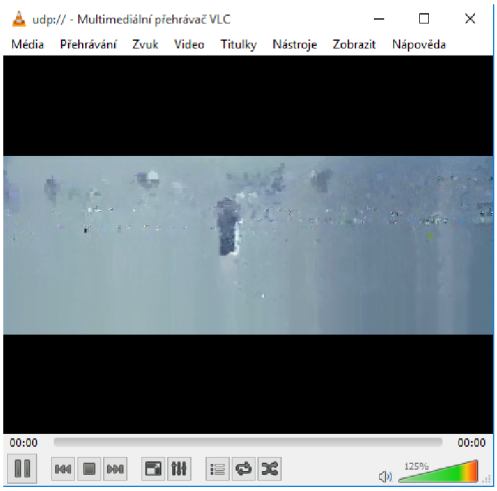
Obrázek 5.9: Video přenos před útokem.

Poté začal útočník odesílat vygenerované pakety, které byly určeny pro stejnou službu na zařízení oběti útoku (Obrázek 5.10). Rychlost útoku dosahovala hodnoty 130 Mb/s.

```
220974 1245.73044 192.168.0.10    192.168.0.11    UDP      1442 Source port: 52566  Destination port: 1234
220975 1245.73044 192.168.0.10    192.168.0.11    UDP      1442 Source port: 52566  Destination port: 1234
220976 1245.73044 192.168.0.10    192.168.0.11    UDP      1442 Source port: 52566  Destination port: 1234
220977 1245.73044 192.168.0.10    192.168.0.11    MPEG TS  1358 Source port: 52566  Destination port: 1234
220978 1245.73044 192.168.0.10    192.168.0.11    UDP      1442 Source port: 52566  Destination port: 1234
220979 1245.73044 192.168.0.10    192.168.0.11    UDP      1442 Source port: 52566  Destination port: 1234
```

Obrázek 5.10: Útok na multimediální datový proud.

Útok byl úspěšný a uživateli na zařízení **PC 1** se začal místo požadovaného video přenosu zobrazovat nesourodý obraz složený z paketů, které dorazily od útočníka. Na obrázku 5.11 je zachycen obraz oběti během útoku.



Obrázek 5.11: Video přenos během útoku.

Obrana před útoky

Jako spolehlivá ochrana před těmito útoky se ukázala brána firewall. Pokud byl tento ochranný prvek aktivní, tak dopady jednotlivých útoků byly dramaticky zredukovány. Ačkoliv došlo k navýšení latence z důvodu zvýšeného provozu, tak brána firewall útoky zablokovala. Nicméně pokud by objem a rychlost přenosu dat byl vyšší jak kapacita přenosové linky, tak ani brána firewall by uživatele neochránila.

V případě útoku SYN flood tyto pakety zcela odfiltrovala. Takové chování u serveru není přípustné, protože by se k serveru nikdo nemohl připojit, a proto je na místě jiný druh ochrany. Jako spolehlivá ochrana se ukázal mechanismus **SYN cookies**. Tento mechanismus funguje na principu vkládání vygenerovaného sekvenčního čísla do hlavičky paketu podle určitých pravidel. Následně žádost vyřadí z fronty a nedochází ke spotřebování cenných prostředků. V případě obdržení odpovědi je na základě sekvenčního čísla schopen určit, zda se jedná o legitimního uživatele.

Kapitola 6

Závěr

Cílem této práce bylo pojednat o síťových útocích Denial of Service, obraně před těmito útoky a vytvoření aplikace pro generování útoků DoS včetně jejího otestování. Všechny cíle bakalářské práce byly splněny ve všech bodech. Zadání konkrétně spočívalo v prostudování problematiky útoků DoS na transportní vrstvě, seznámením se s knihovnou Pcap a následným návrhem a implementací aplikace.

V teoretické části je nastíněna problematika počítačových sítí (Kapitola 2). Konkrétně fungování počítačových sítí, síťová architektura a síťové protokoly. Bez těchto znalostí lze jen těžko pochopit fungování a principy síťových útoků DoS, které jsou popsány v kapitole 3. V této kapitole jsou představeny některé typy útoků DoS a přístupy obrany proti těmto útokům.

V praktické části je popsán návrh aplikace (Kapitola 4). Po získání potřebných teoretických znalostí jsem navrhnul aplikaci, kterou je možné využít k penetračním testům. Aplikace dokáže vygenerovat útoky SYN flood, UDP flood a ICMP flood. V kapitole 5 se věnuji implementaci a testování aplikace. Testování odhalilo, že aplikace plní svůj účel a dokáže vyvinout reálné útoky. Útok SYN flood při testech zahltil server Apache a znepřístupnil službu legitimním uživatelům. Útok ICMP flood zahltil linku oběti velkým množstvím dat. To vedlo k zamezení připojení, zvýšení zatížení procesoru a systém nakonec zcela havaroval. Při testování útoku UDP flood jsem narušil multimediální datový přenos a uživateli se místo požadovaného obrazu zobrazovaly nesourodé pixely. Po zapnutí určitých ochranných prvků se daly dopady těchto útoků zmírnit.

V budoucnu by se na práci mohlo navázat a implementovat další typy útoků. Dále zajistit podporu pro architektury, které využívají princip uložení čísel big-endian. Bylo by také možné rozšířit funkcionalitu a přidat další přepínače pro ovládání útoků.

Literatura

[1] A Cisco Guide to Defending Against Distributed Denial of Service Attacks [online]. http://www.cisco.com/c/en/us/about/security-center/guide-ddos-defense.html, [cit. 2016-05-09].

[2] DDoS Protection [online]. https://cyberwurx.com/ddos-protection/, [cit. 2016-05-09].

[3] TCP 3-Way Handshake [online]. http://okckd.github.io/blog/2014/08/11/Tcp-3way-handshake/, [cit. 2016-05-07].

[4] ITU-T X.200: Open Systems Interconnection - Basic Reference Model: The basic model [online]. `http://www.itu.int/rec/T-REC-X.200-199407-I/en`, 1994-11-07 [cit. 2016-01-16].

[5] Abliz, M.: Internet Denial of Service Attacks and Defense Mechanisms. Technická zpráva, Department of Computer Science, University of Pittsburgh, 2011.

[6] Corero: DDoS Attack Types: Glossary of Terms [online]. `http://www.corero.com/resources/Glossary.html`, [cit. 2016-01-16].

[7] Farrel, A.: *Internet and its protocols – a comparative approach*. San Francisco: Morgan Kaufmann, 2004, ISBN 1-55860-913-X.

[8] Floyd, S.: Congestion Control Principles. RFC 2914, Září 2000.

[9] Gupta, B. B.; Joshi, R. C.; Misra, M.: Distributed Denial of Service Prevention Techniques. *International Journal of Computer and Electrical Engineering*, ročník 2, č. 2, duben 2010: s. 268–276, ISSN 1793-8163.

[10] Lukasik, S.: Why the Arpanet Was Built. *IEEE Annals of the History of Computing*, ročník 33, č. 3, srpen 2011: s. 4–21, ISSN 1058-6180.

[11] Mirkovic, J.: *Internet Denial of Service – Attack and Defense Mechanisms*. New Jersey: Prentice Hall, 2004, ISBN 0-13-147573-8.

[12] Modiri, N.: The ISO reference model entities. *IEEE Network*, ročník 5, č. 4, srpen 2002: s. 24–33, ISSN 0890-8044.

[13] Nagle, J.: Congestion Control in IP/TCP Internetworks. RFC 896, Leden 1984.

[14] Postel, J.: User Datagram Protocol. RFC 768, Srpen 1980.

[15] Postel, J.: Internet Control Message Protocol. RFC 792, Září 1981.

[16] Postel, J.: Transmission Control Protocol. RFC 793, Září 1981.

[17] Shinder, D. L.: *Počítačové sítě*. Praha: SoftPress, 2003, ISBN 80-86497-55-0.