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 APLIKAČNÍ VRSTVĚ

APPLICATION LAYER DOS ATTACK GENERATOR AND PROTECTION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE PAVEL JUHAŇÁK
AUTHOR

VEDOUCÍ PRÁCE Ing. PETR MUSIL
SUPERVISOR

BRNO 2016

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Juhaňák Pavel**
Obor: Informační technologie
Téma: **Generování a ochrana proti DOS útoku na aplikační vrstvě**
Application Layer DOS Attack Generator and Protection
Kategorie: Počítačové sítě

Pokyny:

1. Prostudujte dostupnou literaturu týkající se DOS útoků. Zaměřte se blíže na útoky na aplikační vrstvu.
2. Seznamte se s 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

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato bakalářská práce je zaměřená na generování DoS útoků a ochranu proti nim. Měla by čtenáře uvést do problematiky DoS útoků, objasnit mu jejich dělení a základní principy jejich činnosti. Dále čtenáři nabízí metody a rady, jak se proti těmto útokům bránit, nebo jak zamezit jejich vzniku. Přináší čtenáři náhled na dostupné DoS nástroje a jejich zhodnocení. V rámci této práce je navržen a implementován nástroj pro generování DoS útoků. Ten je schopný generovat HTTP GET, HTTP POST a DNS amplification útoky. Tento DoS nástroj byl úspěšně testován při útocích na webový server Apache 2.2.21. Záznamy a zhodnocení těchto testů jsou také součástí této práce.

Abstract

This bachelor's thesis is focused on generating DoS attacks and protect against them. It should introduce the reader to the problems of DoS attacks, explain their division and the basic principles of their operation to him. Furthermore, it provides methods and advices how to defend against such attacks or to prevent their occurrence. It provides a preview of the available DoS tools and their evaluation to the reader. The tool able to generate DoS attacks is designed and implemented as a part of this work. It is capable of generating HTTP GET, HTTP POST and DNS amplification attacks. The DoS tool was successfully tested during attacks on a web server Apache 2.2.21. The records and evaluation of these tests are also part of this work.

Klíčová slova

DoS, DDoS, RDoS, útok, ochrana, GET útok, POST útok, DNS amplification útok, aplikační vrstva, DoS nástroj

Keywords

DoS, DDoS, RDoS, attack, protection, GET attack, POST attack, DNS amplification attack, application layer, DoS tool

Citace

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

Generování a ochrana proti DOS útoku na aplikační 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.

. . . . . .
Pavel Juhaňák
15. května 2016

Poděkování

Chtěl bych poděkovat vedoucímu práce Ing. Petrovi Musilovi za vydatnou pomoc při vzniku mé práce. Bez jeho odborných rad a připomínek by nemohla vzniknout práce v současné podobě. Také bych rád poděkoval Ludmile Cikrytové a Zdeňce Demelové za jazykové korektury.

Obsah

Kapitola 1

Úvod

Tato bakalářská práce se zabývá generováním DoS (Denial of Service) útoků na aplikační vrstvu a ochranou proti těmto útokům. Útoky jsou vedeny např. proti webovým serverům a jejich účelem je znemožnit serveru poskytování jeho služby. V případě útoku na webový server je cílem znemožnit uživateli prohlížení obsahu (např. webových stránek) tímto serverem poskytovaným. Aby byl systém připojený do internetu odolný proti DoS útokům, je třeba ho zabezpečit a následně jeho ochranu otestovat provedením DoS útoku. Hlavním cílem této práce je tedy navrhnout a implementovat nástroj pro generování DoS útoků na aplikační vrstvu. Tento nástroj bude využit k testování odolnosti proti DoS útokům systémů, vyvíjených v rámci výzkumné činnosti FIT VUT v Brně.

Dalším cílem práce je seznámit čtenáře s tím, co to vlastně je DoS útok a jak je možné rozpoznat, že je na nás veden. Tomuto problému se věnuji ve 2. kapitole společně s rozdělením DoS útoků podle různých kritérií a vysvětlením základních principů činnosti jednotlivých útoků. Po seznámení s problematikou DoS útoků si ve 3. kapitole představíme možnosti ochrany proti nim. Ochrany preventivní, která by útokům měla zabránit, i ochrany aktivní, jež se dostává na řadu při právě probíhajícím útoku. Ve 4. kapitole čtenáře seznámím s historií a principem činnosti nejznámějších nástrojů pro generování DoS útoků. V této kapitole jsou také otestovány a vyjmenovány výhody a nevýhody některých volně dostupných DoS nástrojů.

Specifikaci zadání můžeme nalézt v 5. kapitole. V této kapitole je také popsán návrh mnou implementovaného DoS nástroje – jeho řídící části (jádra) i logiky a principu činnosti jednotlivých útoků. Implementací tohoto nástroje se zabývá 6. kapitola. Především si vysvětlíme hierarchii vzájemného volání metod jednotlivých útoku, ale i implementaci simulačního jádra nebo parsování parametrů. Popis testování a výsledky tohoto testování nalezneme v kapitole číslo 7.

V poslední, tedy 8. kapitole, je provedeno shrnutí činností a výsledků dosažených v této bakalářské práci. Jsou zde navržena možná vylepšení mého DoS nástroje, a to z pohledu efektivity útoku, přenositelnosti a bezpečnosti pro útočníka. Následně je provedeno porovnání mého nástroje s nástroji testovanými ve 4. kapitole. Je zde zhodnoceno, v čem je oproti nim můj nástroj vylepšen.

Kapitola 2

Denial of Service

Denial of Service nebo česky odmítnutí služby, je útok na počítač připojený do internetu navržený tak, aby tomuto počítači znemožnil poskytování jeho služeb a tím zabránil legitimním uživatelům v používání těchto služeb. [1]

Nejčastěji se DoS útoky zaměřují na šířku pásma nebo síťové připojení. Útoky na šířku pásma zaplaví síť tak velkým objemem datových přenosů, že všechny dostupné síťové zdroje oběti jsou spotřebovány, a legitimní požadavky uživatelů již nemohou projít. Útoky na síťové připojení zaplaví počítač tak velkým množstvím požadavků na připojení, že všechny dostupné zdroje operačního systému oběti jsou spotřebovány, a počítač již nemůže zpracovávat legitimní požadavky uživatelů. [23]

Útoky jsou většinou vedeny proti větším společnostem jako například banky, e-shopy, vládní instituce, ale i herní servery apod. Motiv útoku může být odplata, vydírání, hacktivizmus, ale i potěšení z úspěšného útoku či obyčejná zlomyslnost. Dalším motivem může být čistý zisk, např. jedna firma vlastnící e-shop si zaplatí za to, aby konkurenční e-shop byl v období vánoc mimo provoz.

Ne veškerá činnost uživatele vedoucí k odmítnutí služby musí být nutně DoS útok. Například pokud uživatel použije anonymní FTP prostor pro uložení velkého množství dat (ať už dat legálních či nikoliv), konzumuje tím místo na disku a vytváří zvýšený síťový provoz. To může vést k odmítnutí služby a správci onoho serveru se může zdát, že na něj probíhá DoS útok, i když to nebyl uživatelův záměr. [1]

2.1 Příznaky

United States Computer Emergency Readiness Team (US-CERT) definuje příznaky DoS útoku následovně:

- Neobvykle nízký výkon sítě (při otvírání souborů nebo přístupu k webu)
- Nedostupnost konkrétní webové stránky
- Neschopnost přístupu k jakékoliv webové stránce
- Dramatický nárůst počtu nevyžádaných emailů
- Odpojení od bezdrátové sítě nebo kabelového připojení k internetu
- Dlouhodobá nedostupnost přístupu k webu nebo nějakým internetovým službám

DoS útoky mohou vést také k problémům v síťových větvích v okolí počítače, na který je veden útok. Například pokud by byla maximální šířka pásma směrovače spojující lokální síť s Internetem zkonzumována útokem, pak by byl útokem zasažen nejen počítač, který byl cílem, ale také všechny ostatní počítače v místní síti, popřípadě celá firemní síť. [1]

2.2 Typy útoků

DoS útoky se objevují v mnoho formách a jsou zaměřeny na různé služby. Zde jsou tři základní typy DoS útoků:[1]

- Spotřeba nedostatkových, omezených nebo neobnovitelných zdrojů
- Poškození nebo úprava konfiguračních nastavení
- Fyzické poškození nebo pozměnění prvků sítě

Spotřeba omezených zdrojů

Počítače a sítě potřebují k provozu jisté věci neboli zdroje, jimiž jsou například: šířka pásma, operační paměť, místo na disku, procesorový čas, datové struktury, přístup k jiným počítačům a sítím, ale také některé fyzické (přírodní) zdroje jako jsou elektřina, studený vzduch nebo voda.

Připojení k síti

Nejčastějším cílem DoS útoků se stává připojení k síti. Cílem je zabránit uživatelům nebo celým sítím v komunikaci po síti. Zástupcem tohoto typu útoků je například „SYN flood“ útok.

V tomto typu útoku útočník započne proces navázání spojení s počítačem oběti, dělá to však takovým způsobem, že nikdy nedojde k dokončení navázání tohoto spojení. V tomto případě počítač oběti rezervuje jednu z omezeného počtu datových struktur potřebných pro navázání spojení. Výsledkem je, že dojde k vyčerpání těchto datových struktur, což zapříčiní, že legitimní připojení jsou odmítána, protože počítač oběti čeká na dokončení falešných polo–otevřených připojení.

Měli bychom si uvědomit, že tento typ útoku nezávisí na útočníkově schopnosti spotřebovat šířku pásma sítě oběti. V tomto případě útočník spotřebovává datové struktury jádra systému zajišťující síťové připojení. Z toho vyplývá, že útočník může úspěšně útočit i z mnohem pomalejší sítě, než jakou používá oběť. Toto je dobrý příklad asymetrického útoku.

Použití Vašich zdrojů proti Vám

Útočník může neočekávanými způsoby použít Vaše vlastní zdroje proti Vám. Příkladem tohoto typu útoků je „UDP packet storm“ útok. Při tomto útoku útočník použije podvrhnuté UDP packety k připojení echo služby na jednom počítači k chargen (character generator) službě na počítači druhém. Výsledkem je vyčerpání veškeré volné šířky pásma sítě mezi těmito dvěma službami. To znamená, že připojení k síti všech počítačů nacházejících se ve stejné síti jako jeden z počítačů, na něž je veden útok, bude ovlivněno.

Spotřebování šířky pásma

Útočník může být schopen spotřebovat veškerou volnou šířku pásma sítě, na níž útočí tím, že generuje velké množství packetů směřujících do této sítě. Typicky se jedná o ICMP ECHO packety, tzv. „Ping flood“ útok, ale v zásadě lze použít jakékoliv packety. Útočník také nemusí útočit pouze z jednoho stroje a jedné sítě. Tím, že zaútočí z více sítí, sníží požadavek na šířku pásma připojení jednotlivých jím ovládaných počítačů. Protože tento typ útoku je symetrický, útok může být úspěšný, pouze pokud útočník disponuje větší celkovou šířkou pásma, než jeho oběť.

Spotřebování ostatních zdrojů

Kromě šířky pásma sítě může být útočník schopen spotřebovat i jiné zdroje, které systém k provozu potřebuje. Například v mnoha systémech je omezený počet datových struktur pro uchování informací o procesech (identifikátory procesů, tabulky procesů, atd.). Útočník může být schopný spotřebovat tyto datové struktury pomocí jednoduchého programu nebo skriptu, který nebude dělat nic jiného, než sám sebe kopírovat, tzv. „Fork bomb“ útok. Mnoho moderních operačních systémů je proti tomuto útoku chráněno, např. omezený počet vytvořených vláken/procesů jedním procesem, ale ne všechny nějakou ochranu používají. Nicméně i když není tabulka procesů zcela zaplněna, může být počítač značně zpomalen a to tím, že pokud je otevřen příliš velký počet procesů, procesor může být zahlcen režií spojenou s přepínáním kontextu mezi jednotlivými procesy.

Útočník se také může pokusit spotřebovat diskový prostor, např. následujícími způsoby:

- Generování nadměrného množství emailů
- Úmyslné generování chyb, které musí být zaznamenány
- Ukládání souborů na anonymní FTP prostor nebo síťová úložiště

Obecně platí, že jakákoliv služba umožňující zapisovat data na pevný disk, může být zneužita pro DoS útok, pokud nemá nastavena nějaká pravidla omezující maximální velikost zapsaných dat pro jednoho uživatele.

Také mnoho webových stránek nebo serverů používá systém uzamčení účtu na stanovenou dobu při určitém počtu neúspěšných přihlášení. Typicky systém účet uzamkne po třech až pěti neúspěšných pokusech o přihlášení. Toto opatření slouží jako ochrana proti takzvanému „brute–force cracking“ útoku, při němž se útočník pomocí nějakého skriptu snaží prostým zkoušením jedné kombinace za druhou uhodnout heslo k účtu. Útočník tohoto opatření může využít a zabránit tak legitimním uživatelům v přihlášení tím, že se bude neustále pokoušet přihlásit k různým účtům se špatným heslem. V některých případech tomuto útoku mohou podlehnout i privilegované účty, jako jsou root nebo administrator. Proto je dobré, aby měl administrátor nějakou jinou metodu získání přístupu do systému, v případě těchto mimořádných okolností. Tyto metody získání přístupu je vhodné konzultovat s výrobcem operačního systému.

Útočník může být schopen zapříčinit nestandardní chování nebo dokonce pád systému tím, že mu po síti pošle nějaká neočekávaná data. Příkladem tohoto typu útoku je „Ping of death“ při němž útočník zasílá oběti příliš velké ICMP datagramy, což může způsobit selhání, zamrznutí nebo restart sytému. V případě, že u systému zaznamenáváme časté pády bez zjevné příčiny, mohou být způsobeny právě tímto útokem.

Kromě počítačů jsou zde i jiná zařízení, na která se může útočník změřit a způsobit u nich odmítnutí služby. Kromě samotných počítačů jsou nejčastějším terčem DoS útoků

směrovače, přepínače, síťová úložiště a síťové tiskárny. Nicméně cílem útoku se může stát jakékoliv zařízení připojené k síti.

Poškození nebo úprava konfiguračních nastavení

Nesprávně nakonfigurovaný počítač nemusí dobře fungovat, nebo nemusí fungovat vůbec. Útočník může být schopný poškodit či upravit konfigurační soubory počítače nebo jiných síťových zařízení a tím negativně ovlivnit fungování těchto zařízení.

Například, pokud dokáže útočník změnit nebo ovlivnit obsah směrovací tabulky na směrovači, bude komunikace s touto sítí ochromena. Toto je následkem například útoku známého jako „RIP spoofing“, při němž útočník naslouchá zprávám protokolu RIP a následně tyto zprávy podvrhne, čímž změní obsah směrovacích tabulek na směrovačích v síti. Pokud útočník dokáže upravit hodnoty v registrech na zařízeních s operačním systémem Windows NT®, mohou mít některé funkce nestandardní nebo chybné chování, popřípadě mohou být úplně nedostupné.

Fyzické poškození nebo pozměnění prvků sítě

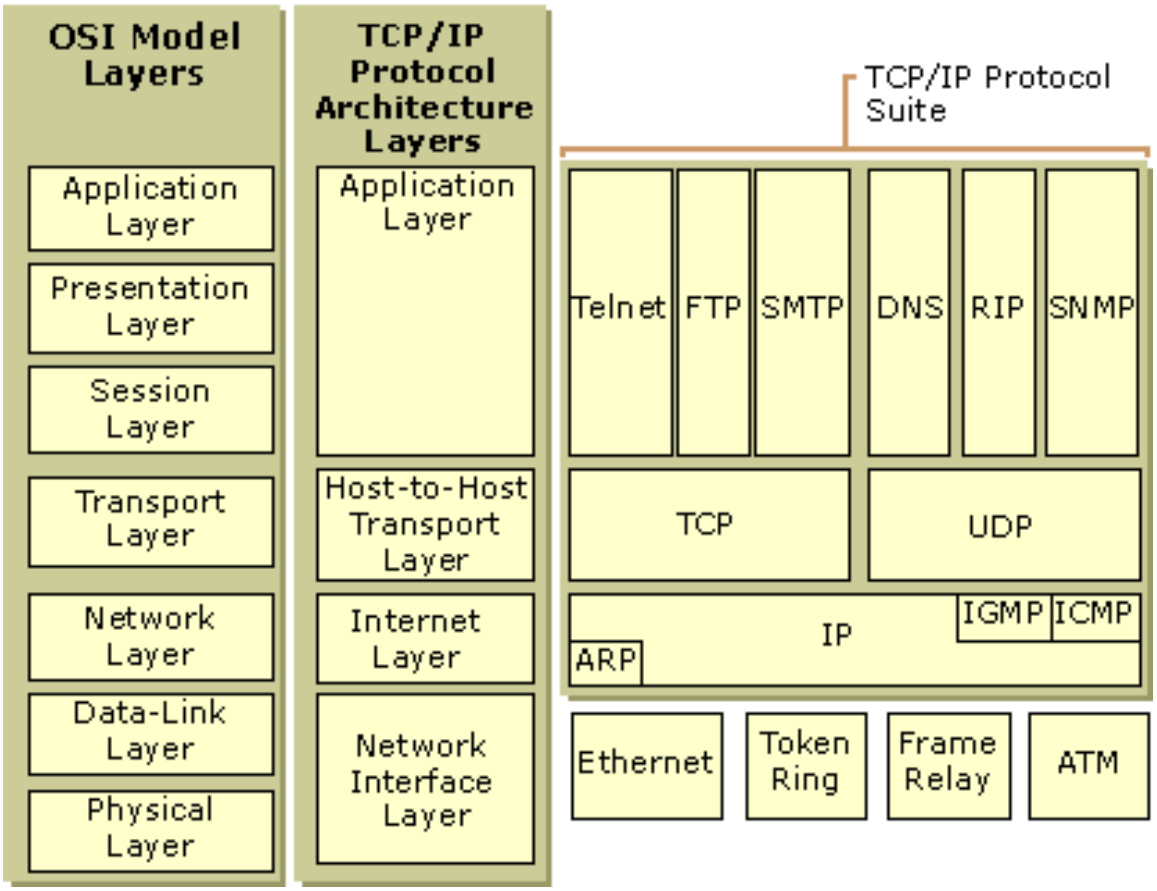
Hlavním problémem u tohoto typu útoku je fyzické zabezpečení. Proti neautorizovanému přístupu bychom měli chránit počítače, servery, směrovače, rozvodné (propojovací) skříně, prvky páteřní sítě, elektrické a chladící stanice a další komponenty kritické pro chod sítě či elektronických zařízení obecně.

Fyzická bezpečnost je hlavním prvkem v ochraně proti těmto typům DoS útoků. Fyzickou bezpečnost by měly zajišťovat zákony dané země a policejní složky dohlížející na dodržování těchto zákonů. Policisté však nemohou být vždy všude, proto je dobré najmout si bezpečnostní agenturu a zavést politiku omezeného přístupu do klíčových částí budovy. Kabely páteřní sítě jsou proti poškození chráněny umístěním pod zem nebo na dno oceánu. Nicméně ani toto opatření není vždy dostatečné, jak ukazuje kauza důchodkyně, která při hledání mědi přerušila optické vedení páteřní sítě a odstřihla tak od internetu Arménii a část Gruzie. [14]

2.3 Dělení útoků podle ISO/OSI

Typy útoků můžeme dále dělit podle vrstvy ISO/OSI modelu, na kterou útočí. ISO/OSI model je referenční komunikační model, který rozděluje komunikaci po síti do 7 vrstev modelu. Nižší vrstvy poskytují vyšším vrstvám abstrakci komunikace. Tedy např. aplikační vrstva na jednom PC chce komunikovat s aplikační vrstvou na druhém PC a nezajímá ji, jak přesně se tato komunikace provede. K tomu využije služby prezentační vrstvy, ta bude využívat služby nižší vrstvy atd. až k vrstvě fyzické. Fyzická vrstva je jediná, která skutečně propojuje dva počítače mezi sebou. ISO/OSI model nenabízí přímo implementaci funkcí jednotlivých vrstev, ale pouze architekturu a popis vrstev a jejich funkcí. V kontextu vrstev ISO/OSI modelu jsou útoky vedeny především na aplikační a transportní vrstvu.

Na obrázku 2.1 můžeme vidět architekturu sedmi vrstvého ISO/OSI modelu v porovnání s čtyř vrstvým TCP/IP modelem. Také je zde znázorněno, na jakých vrstvách pracují některé známé protokoly. V následujících podsekcích jsou stručně popsány funkce jednotlivých vrstev a typické útoky na tyto vrstvy.



Obrázek 2.1: OSO/OSI model, TCP/IP model a protokoly podle TCP/IP modelu [6]

(L1) Fyzická vrstva

Fyzická vrstva specifikuje fyzickou komunikaci. Aktivuje, udržuje a deaktivuje fyzické spoje mezi koncovými systémy. Do fyzické vrstvy patří např. opakovače, kabely a konektory. Mezi protokoly používané fyzickou vrstvou patří např. 100Base-T. [4] [11]

Při útoku na tuto vrstvu se útočník snaží fyzicky zničit nebo poškodit nějaké zařízení pracující na této vrstvě a tím znemožnit síťovou komunikaci, než bude toto zařízení vyměněno nebo opraveno. [4]

(L2) Linková vrstva

Linková vrstva uspořádává data z fyzické vrstvy do logických celků známých jako rámce. Řadí přenášené rámce, stará se o nastavení parametrů přenosu linky, oznamuje neopravitelné chyby. Formátuje fyzické rámce, opatřuje je fyzickou adresou (MAC) a poskytuje synchronizaci pro fyzickou vrstvu. Na linkové vrstvě pracují síťové karty, přepínače a přístupové body (AP). Mezi protokoly používané linkovou vrstvou patří např. 802.3 a 802.5. [4] [11]

Příkladem útoku na linkovou vrstvu je „MAC flood“ útok, při kterém útočník posílá datové packety s různými MAC adresami, čímž zaplní RAM paměť nebo paměťový prostor vyhrazený pro tabulku MAC adres. Tím zapříčiní buď nestabilitu či pád systému nebo

alespoň zabrání připojení nových uživatelů, tedy spíše uživatelů s MAC adresou, která ještě není v tabulce. [4]

(L3) Síťová vrstva

Tato vrstva se stará o směrování v síti a síťové adresování. Obsahuje funkce, které umožňují překlenout rozdílné vlastnosti technologií v přenosových sítích. Síťová vrstva poskytuje funkce k zajištění přenosu dat různé délky od zdroje k příjemci skrze jednu případně několik vzájemně propojených sítí. Na této vrstvě pracují směrovače. Nejznámějším protokolem pracujícím na síťové vrstvě je protokol IP. Mezi další protokoly používané síťovou vrstvou patří např. ICMP, ARP, RIP. [4] [11]

Příkladem útoku na síťovou vrstvu je „ICMP flood“ útok, což je společný název pro útoky s podobným principem, jako jsou např. „Ping flood“, „Ping of death“. Tyto útoky používají protokol ICMP k spotřebování šířky pásma sítě. [4]

(L4) Transportní vrstva

Tato vrstva zajišťuje přenos dat mezi koncovými uzly. Jejím účelem je poskytnout takovou kvalitu přenosu, jakou požadují vyšší vrstvy. Vrstva nabízí spojově (TCP) a nespojově (UDP) orientované protokoly. Protokoly TCP a UDP jsou vyžívány při použití architektury TCP/IP, kromě těchto na transportní vrstvě pracují např. protokoly IPX, SNA. [4] [11]

Příkladem útoků na síťovou vrstvu jsou „SYN flood“ a „Smurf Attack“. Cílem těchto útoků je spotřeba šířky pásma sítě nebo omezený počet datových struktur pro navázání spojení, tedy dosažení maximálního počtu aktivních připojení na počítači nebo některém síťovém zařízení oběti, čímž je jí znemožněna komunikace po síti. [4]

(L5) Relační vrstva

Smyslem vrstvy je organizovat a synchronizovat dialog mezi spolupracujícími relačními vrstvami obou systémů a řídit výměnu dat mezi nimi. Umožňuje vytvoření a ukončení relačního spojení, synchronizaci a obnovení spojení, oznamovaní výjimečných stavů. Na této vrstvě se používají například protokoly Logon/Logoff, NFS, NetBios. [4] [11]

Příkladem útoku na relační vrstvu je „Telnet DDoS“ útok, který využívá chyb v softwaru Telnet serverů běžících na síťových prvcích, čímž službu Telnet vyřadí z provozu. Vyřazením služby Telnet útočník znemožní správci přihlásit se k síťovému zařízení, a tím mu znemožní jeho údržbu či konfiguraci. [4]

(L6) Prezentační vrstva

Funkcí vrstvy je transformovat data do tvaru, který používají aplikace (šifrování, konvertování, komprimace). Formát dat se může lišit na obou komunikujících systémech, navíc dochází k transformaci pro účel přenosu dat nižšími vrstvami. Používá protokoly pro kompresy a šifrování, jako např. SSL, NDR, ICA. [4] [11]

Příkladem útoku na prezentační vrstvu je „Malformed SSL requests“ útok. Kontrola SSL šifrování příchozích packetů je samo o sobě náročné na systémové zdroje. Útočník toho využívá a zasílá chybné SSL požadavky, které jsou ještě náročnější na zpracování, čímž vyčerpá systémové zdroje. To způsobí, že server přestane přijímat SSL připojení, nebo se automaticky restartuje. Znemožní tak přístup k HTTP serveru. [4]

(L7) Aplikační vrstva

Účelem vrstvy je poskytnout aplikacím přístup ke komunikačnímu systému a umožnit tak jejich spolupráci. Na této vrstvě začíná tvorba zpráv a packetů. Identifikují se zde komunikující strany, je zde nastavována požadovaný kvalita služeb (QoS), jsou dne ověřováni uživatelé a nastavovány požadavky a pravidla pro bezpečnost a soukromí komunikace. Veškerá činnost na této vrstvě je aplikačně specifická. Do této vrstvy se řadí například tyto služby a protokoly: FTP, SMTP, Telnet, DNS, DHCP, SSH. [4] [11]

Příkladem útoků na aplikační vrstvu jsou útoky zaměřené na webové servery, jako jsou například „PDF GET“, „HTTP GET“, „HTTP POST“ útoky, jimiž je možno zneužít webové formuláře k útoku. Útočník může například posílat velký soubor (několik GB) a to velmi pomalu (např. 1Byte za minutu). Tím zabírá jeden slot připojení a taky alokovanou paměť (RAM nebo diskovou) pro příjem onoho souboru. Pokud takovýchto souborů takto pomalu začne posílat dostatečné množství, serveru paměť (RAM nebo disková) dojde. Může způsobit restart, nebo alespoň odmítnutí dalších uživatelů. Útočník může také otevírat nová připojení k webovému serveru a udržovat je otevřená. Pokud jich otevře dostatečné množství, zabrání tím v otevření nových spojení nově příchozím legitimním uživatelům. [4]

2.4 Další dělení útoků

U DoS útoků můžeme rozlišovat, zda je před samotným útokem potřeba ustanovit spojení, nebo ne. Dále rozlišujeme, zda útočník vede útok ze svého počítače přímo na počítač oběti, zda při útoku požívá prostředníka, tzv. odražený útok, nebo k útoku používá celou síť počítačů, tzv. distribuovaný útok. Toto rozdělení bude popsáno v následujících podkapitolách.

Spojově orientované útoky

Před provedením samotného útoku je u spojově orientovaných služeb potřeba nejdříve ustanovit spojení. Až po legitimním připojení k oběti je možno provádět útok. Spojově orientované útoky jsou vedeny především na služby pracující nad protokolem TCP, který je spojově orientovaný. Zástupci těchto útoků jsou útoky na protokol HTTP, např. „HTTP GET“ útok.

Následkem nutnosti provést legitimní připojení k oběti je, že oběť zná identitu (IP adresu) útočníka, což je nevýhodou. Proto se spojově orientované útoky často uskutečňují za pomoci botnetu, nebo se pro připojení k oběti používají proxy servery. Tím zůstane zachována anonymita útočníka.

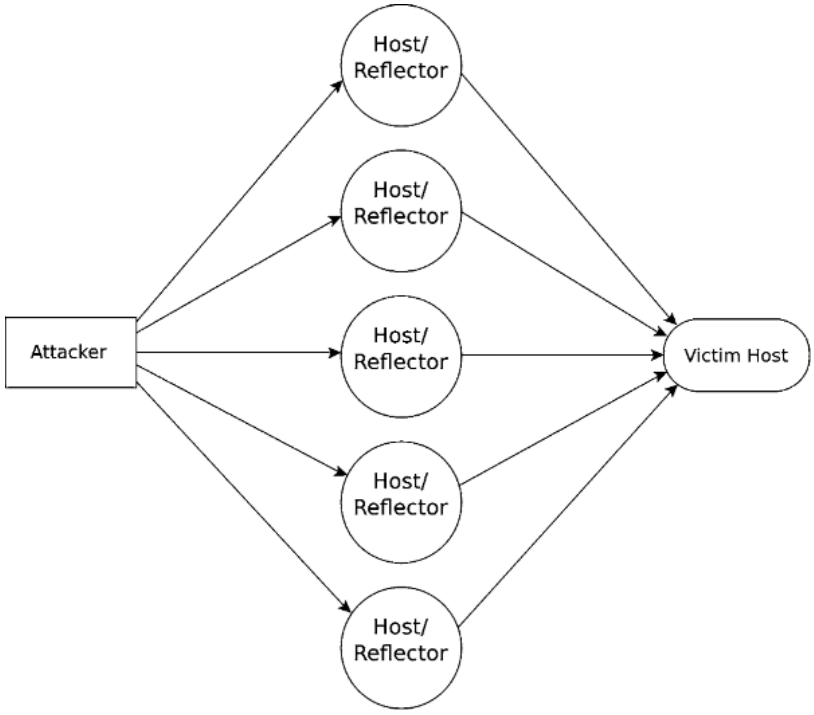
Nespojově orientované útoky

Před provedením tohoto typu útoků není potřeba nijak komunikovat s obětí, prostě rovnou zahájíme útok. Nespojově orientované útoky jsou převážně útoky na šířku pásma. A to buď přímo, např. „ICMP flood“, nebo odražením přes server hostující službu pracující nad protokolem UDP, např. „DNS Amplification Attack“. Nespojově orientovaným útokem můžeme útočit i na spojově orientovanou službu a to tím, že využijeme její spojovosti proti ní. Příkladem je útok na protokol TCP, konkrétně útok „SYN flood“.

Výhodou nespojově orientovaných útoků je, že lze snadno podvrhnou IP adresy a tím chránit identitu útočníka. U odražených útoků je anonymita útočníka zachována implicitně.

Odražené útoky

Technika odraženého, nebo-li reflektovaného (RDoS), útoku spočívá v tom, že útočník neútočí přímo na svou oběť, ale k útoku používá prostředníka, většinou však větší množství prostředníků. Na tyto počítače útočník posílá požadavky s podvrhnutou zdrojovou IP adresou. Zdrojovou IP adresu nastaví na IP adresu cíle svého útoku. Počítače, které požadavek obdrží, na něj odpoví, a všechny odpovědi směřují na počítač oběti, čímž dojde k vyčerpání zdrojů, na které je útok zaměřen (šířka pásma, paměť, atd.). Takto lze útočit pouze na nespojově orientované služby, tedy především na služby pracující nad UDP. Výhodou tohoto typu útoku je, že útočník zůstává anonymní, protože oběť nezná IP adresu útočníka. Příkladem tohoto útoku jsou „DNS Amplification Attack“ a „Smurf Attack“. Na obrázku 2.2 můžeme vidět schéma odraženého útoku. [3] [20]



Obrázek 2.2: Schéma odraženého (RDoS) útoku [24]

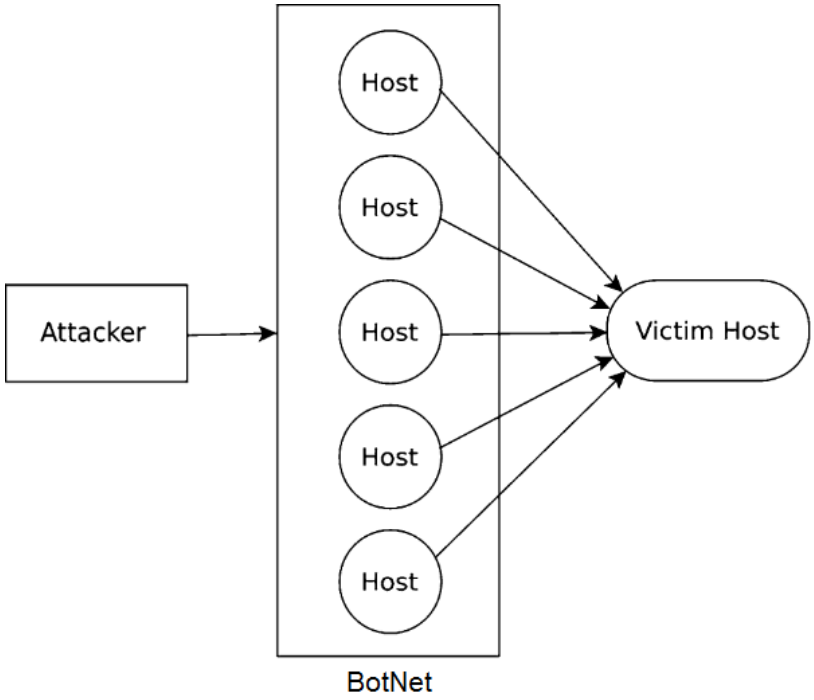
Zesílené útoky

S technikou odrážení útoku popsané v předchozí podkapitole se velmi často kombinuje technika zesílení útoku. Tato technika je založena na skutečnosti, že při zaslání dotazu na určitou službu je odpověď oné služby několikrát větší, než byl původní dotaz. Služby s největším poměrem zesílení vůči dotazu jsou NTP (cca 550x), CharGen (cca 350x) a DNS (cca 50x). Útočník v tomto případě vede útok odrazem od několika serverů vybrané zesilující služby, čímž dojde k vyčerpání zdrojů, na něž je útok zaměřen. Díky zesilujícímu efektu nemusí být útok odrážen od desítek počítačů, ale stačí ho odrazit od několika vhodných serverů. Tento útok také musí být veden pouze přes nespojově orientované služby a přes servery, které nevyžadují autentizaci. [3] [20]

Distribuované DoS útoky

Distribuovaný DoS útok (DDoS) je jednou z variant běžného DoS útoku. Klíčovým rysem DDoS útoků je obrovský počet uživatelů, nebo spíše jejich počítačů, použitých k uskutečnění tohoto útoku. Je běžné, že k provedení DDoS útoku jsou využity stovky tisíc až miliony počítačů. Často jsou tyto počítače k DoS útokům používány bez vědomí jejich majitelů. Stává se tak z důvodu nakažení počítače virem, trojským koněm nebo do něj byl nainstalován program se skrytou funkčností. Všechny tyto druhy nakažení umožňují útočníkovi na dálku ovládat počítač a využívat ho k provádění DoS útoků. Tyto počítače se označují jako „boti“ nebo zombie a skupina takovýchto počítačů ovládaných jedním útočníkem se nazývá „botnet“. [24]

„Botnety“ jsou často využívány útočníkem nebo skupinou útočníků, jako nástroj k provádění vysoce výkonných útoků na šířku pásma. Nicméně nemusí se jednat jen o tento typ útoků, „botnety“ bývají využívány také pro šíření spamu, distribuci malwaru nebo odchytávání síťové komunikace. K provedení útoku za pomoci „botnetu“ může útočník použít jednu nebo kombinaci více technik útoků probíraných výše, tedy některý z klasických DoS útoků, odražený nebo zesílený útok. Techniky zesílení útoku používá útočník k zesílení už tak velmi silného DDoS útoku. Díky využití „botnetu“ je odhalení identity útočníka velice nepravděpodobné, přesto může používat techniky odraženého útoku k chránění identity jednotlivých „botů“, protože bez „botů“ není „botnet“. Pokud se tyto techniky kombinují, velmi rychle spotřebují dostupnou šířku pásma oběti. Proti tomuto útoku je velice těžké se bránit. Vzhledem k tomu, že počítače zapojené v „botnetu“ mohou být kdekoliv na světě, což znamená, že útok může přicházet z mnoha různých sítí zároveň, je preventivní i aktivní ochrana velice náročná. Na obrázku 2.3 můžeme vidět schéma DDoS útoku.



Obrázek 2.3: Schéma distribuovaného (DDoS) útoku [24]

Kapitola 3

Ochrana proti DoS útokům

Jak jsme si ukázali v přechozí kapitole, existuje celá řada DoS útoků a to jsme zmínili pouze jejich základní dělení a některé významné zástupce těchto útoků. S přibývajícím počtem druhů a implementací jednotlivých útoků je třeba věnovat zvýšenou pozornost a úsilí ochraně proti nim. Ve prospěch útočníků bohužel hraje to, že nikdy nemůžeme předem vědět, kterým z oněch útoků se rozhodne zaútočit. Proto bychom v ideálním případě měli implementovat ochranu proti všem druhům útoků. To ovšem pro klasickou firmu nemusí být reálné, protože by to znamenalo najmout team bezpečnostních techniků, kteří by navrhli a implementovali všechny tyto ochrany, udržovali je a aktivně sledovali trendy a nové metody útoků a vylepšovali by stávající metody ochrany, popřípadě navrhovali nové tak, aby byly schopny chránit proti těmto útokům. Naštěstí se firmy nemusejí spoléhat pouze na sebe, ale existují společnosti zabývající se ochranou proti DoS/DDoS útokům a nabízející komplexní ochranu proti téměř všem typům útoků. Takovouto společností je např. společnost Imperva a její systém Incapsula[1], která podle jejich webových stránek chrání proti všem typům útoků, kromě specifických útoků na aplikační vrstvu.

Druhy ochrany proti DoS útokům bych rozdělil na dvě skupiny a to ochranu pasivní (preventivní) a ochranu aktivní. Tyto druhy ochrany popíšu v následujících podkapitolách.

3.1 Pasivní (preventivní) ochrana

Tento typ ochrany zahrnuje nastavení služeb, konfiguraci protokolů, popřípadě nástavby nebo přídavné funkce protokolů a služeb. Tato opatření jsou aktivní neustále a pro všechny uživatele služby. Úpravou nastavení se snažíme o to, aby útok nebyl vůbec možný, nebo byl jeho dopad co nejnižší. [1] [4] [19] [24]

Pro ochranu proti mnou implementovanému „HTTP GET“ útoku je možno upravit následující nastavení:

- Co nejvíce zvýšit počet současně připojených uživatelů. Tím pádem bude muset útočník otevřít více připojení, pokud bude chtít obsadit všechny volné sloty pro připojení.
- Omezit počet packetů v nichž lze serveru předávat HTTP GET požadavek. Je třeba vysledovat průměrný počet packetů na jeden požadavek a nastavit strop o něco vyšší. Útočník se bude muset vypořádat se serverem generovanými chybami a znovu navazovat útok.

[1] https://www.incapsula.com/ddos-protection-services.html

- Snížit maximální čas, v němž musí server obdržet další část HTTP GET požadavku. Pro správné nastavení intervalu je třeba opět zjistit průměrný čas doručení další části požadavku a limit nastavit o něco vyšší. Díky tomu bude muset útočník posílat další část požadavku častěji a tím se zvýší vytížení i jeho stroje.

Pokud budeme schopni obsloužit dostatečné množství připojených uživatelů současně a budeme požadovat další část hlavičky v dostatečně malém intervalu, je možné, že útočník nebude schopen generovat takový počet připojení s dostatečně malými intervaly mezi částmi požadavku. Tím pádem bude jeho útok neúčinný a navíc zatíží svůj vlastní stroj.

Pro ochranu proti mnou implementovanému „HTTP POST“ útoku je možno upravit následující nastavení:

- Zmenšit maximální velikost nahrávaného souboru a maximální velikost odesílaného HTTP POST požadavku. Útočník tak bude muset častěji navazovat útok.

- Omezit maximální počet nahraných souborů pro jednotlivé uživatele, např. podle IP adresy. Útočník tak nebude moci systém přetížit nahráním příliš velkého počtu souborů.

- Snížit maximální čas, v němž musí server obdržet další část posílaného souboru. Pro správné nastavení intervalu je třeba zjistit průměrný čas doručení další části souboru a limit nastavit o něco vyšší. Díky tomu bude muset útočník posílat další část souboru častěji, čímž zkrátí čas blokace slotu připojení.

Pokud omezíme maximální velikost souboru a maximální počet odeslaných souborů společně s omezením maximálního času pro přijetí další části souboru, útočník za nepříliš dlouhý čas odešle onen počet souborů a dále nebude blokovat sloty připojení, čímž jeho útok skončí neúspěchem.

Pasivní ochrana proti „DNS Amplification Attack“ na straně oběti není možná, proto je třeba vsadit na prevenci proti útokům na straně DNS serverů. Veřejné DNS servery by neměly umožňovat rekurzivní vyhledávání. Výrazně se tak zmenší velikost odpovědi serveru a takovýto DNS server se nedá použít k útoku. Pokud si přejeme provádět rekurzi např. na firemním DNS serveru, který je zároveň serverem autoritativním, tedy dostupným veřejnosti, je třeba vytvořit „access control list“ jímž omezíme provádění rekurzního vyhledávání pouze pro definované klienty, např. klienty z místní sítě.

3.2 Aktivní ochrana

Tento typ ochrany upravuje chování systému poté, co systém detekuje, že se nachází pod DoS útokem. Opatření stanovená tímto typem ochrany jsou platná pouze pokud je na systém veden útok a ideálně zasahují pouze útočníka, přičemž nemají vliv na standardní uživatele. [4] [8] [19] [24]

Pro aktivní ochranu je velmi důležité zjištění, že je na systém nebo službu veden DoS útok. Toho lze docílit měřením a analýzou statistik síťového provozu, počtu požadavků na službu, počtu současně připojených uživatelů apod. Tato data se zpětně analyzují, zjišťují se průměrné hodnoty a poté se porovnávají s hodnotami naměřenými v reálném čase. Pokud některá ze sledovaných hodnot výrazně přesáhne průměrné hodnoty, je systém pravděpodobně pod DoS útokem. Sledování všech možných ukazatelů by bylo výpočetně velmi náročné, proto je rozumné zvolit pouze několik z nich. Jako minimální vhodnou kombinaci bych zvolil sledování vytíženosti šířky pásma a počet současně připojených uživatelů.

Tato kombinace odhalí jak útoky na šířku pásma např. „DNS Amplification Attack“, tak i takzvané pomalé útoky např. „HTTP GET“ útok.

Po zjištění, že je systém pod útokem, je dalším krokem určit, kdo na něj útočí. Toho docílíme podrobnější analýzou příchozích dat v reálném čase. Protože tato analýza může být výpočetně náročná, neprovádíme ji neustále, ale pouze pokud máme podezření, nebo jsme si jisti, že je systém pod DoS útokem. Pokud je na nás veden jeden z tzv. pomalých útoků, můžeme analýzu provádět na stroji, na který je veden útok. Pokud jsme ovšem pod útokem na šířku pásma, je vhodné analýzu provádět na jiném stroji, nebo dokonce rozdělit příchozí datový tok na několik menších a na těch provádět analýzu paralelně několika stroji zároveň. Pokud nemáme takové technické možnosti, můžeme vybírat analyzovaná data náhodně z příchozího datového toku, nebo provádět analýzu postupně pro jednotlivé připojené uživatele.

V následující části této sekce budu popisovat některé ukazatele na útočníka, tedy různé faktory, jejichž porušením se útočník prozrazuje. Mějme prosím na paměti, že hodnoty, které uvádím, jsou pouze ukázkové a nemusí být platné ve všech případech. Pro přesné určení těchto hodnot je třeba předem provést analýzu síťového toku systému při bezproblémovém provozu a tyto výsledky uchovat právě pro porovnání s hodnotami naměřenými v reálném čase při útoku.

Pro zjištění původu „HTTP GET“ útoku je možno sledovat následující ukazatele:

- Počet aktivních připojení navázaných jedním uživatelem. Klasický uživatel si například otevře do deseti aktivních oken naší stránky, ale rozhodně si jich neotevře desítky, nebo dokonce stovky.

- Počet přijatých packetů pro dokončený HTTP GET požadavek. U klasických webových stránek stačí jeden packet, na jeden HTTP GET požadavek. Proto pokud jsme obdrželi desítky packetů a požadavek ještě není dokončený, jedná se pravděpodobně o jedno z útočících připojení.

- Časové rozložení mezi příchozími packety. V reálném světě přichází síťová komunikace s exponenciálním rozložením. Pokud packety od některého z uživatelů přicházejí s výrazně jiným rozložením, jako např. s konstantním, nebo s rozložením exponenciálním, jehož parametr střední hodnoty je výrazně vyšší, než průměrný parametr střední hodnoty, jedná se pravděpodobně o útočníka.

- Velikost, obsah a validita požadavku. Obsah a velikost požadavků je značně odlišná u každé provozované služby. Obecně můžeme říci, že pokud obdržíme několik nevalidních po sobě jdoucích HTTP GET požadavků, jedná se pravděpodobně o jedno z útočících připojení.

Pro zjištění původu „HTTP POST“ útoku je možno sledovat stejné ukazatele, jako u „HTTP GET“ útoku, a také následující ukazatele:

- Počet souborů nahraných jedním uživatelem za určitý časový úsek. Klasický uživatel nahraje např. méně než 5 souborů denně.

- Rozložení velikosti a velikost dat jednotlivých částí souboru. Klasický uživatel bude posílat data po několika kilobitech, naproti tomu útočník je bude posílat jen po několika málo bitech. Rozložení velikosti dat závisí na aplikaci.

- Časové rozložení mezi přijetím jednotlivých částí souboru. Podobně jako u časového rozložení mezi příchozími packety očekáváme exponenciální časové rozložení s průměrně velkým parametrem střední hodnoty.
- Velikost a obsah souboru. Pokud příjmene od jednoho uživatele několik velkých, nečitelných, poškozených nebo textových souborů s náhodných obsahem, jedná se pravděpodobně o útočníka.

Pro zjištění původu „DNS Amplification Attack“ je možno zastávat postoje, že jakákoliv DNS odpověď, kterou přijme např. webový server a nepochází z firemního DNS serveru, je považována za útok. Pokud nechceme takto striktní postoj, nebo naše firma neprovozuje a nepřeje si provozovat DNS server, můžeme sledovat následující ukazatele:

- Počet příchozích DNS odpovědí z jednoho DNS serveru za určitý časový úsek. Při klasickém provozu např. méně než 10 přijatých DNS odpovědí za minutu.
- DNS odpovědi na stejné webové stránky a to jak z jednoho serveru, tak mezi více různými DNS servery. Útočník může posílat stejný dotaz, nebo několik dotazů pořád dokola na jeden nebo více DNS serverů.
- Časové rozložení mezi přijatými odpověďmi z DNS serveru. Podobně jako v předchozích případech, očekáváme exponenciální časové rozložení s průměrně velkým parametrem střední hodnoty.
- Příchozí odpověď bez předchozího požadavku. Je možné sledovat odeslané DNS požadavky a pokud přijde odpověď na požadavek, který nebyl odeslán, jedná se pravděpodobně o útok.

Pokud jsme některou ze zmíněných metod odhalili adresu útočníka, nebo více adres, které jsou zdrojem útoku, přidáme pro ně blokovací pravidlo do firemního firewallu. V případě, že je na nás veden některý z tzv. pomalých útoků, mělo by toto opatření postačovat. Pokud je na nás veden útok na šířku pásma, je možné, že firewall takové množství dat nezvládne filtrovat. Tím se účinek DoS útoku přesune ze serveru, na který byl původně veden útok, na firewall. Řešením může být podobně jako u analýzy dat rozdělení příchozího datového toku na několik menších a ty paralelně filtrovat přes několik firewallů zároveň. Po odfiltrování nežádoucích packetů jejich opětovné spojení do jednoho datového toku. Další možností může být blízká spolupráce s ISP. Pokud identifikujeme zdroj útoku a nejsme schopni útok zastavit sami, kontaktujeme ISP, předáme mu informace o probíhajícím útoku a podklady k jeho zastavení. [19]

Pokud nejsme schopni určit původ útoku, nebo je útok příliš silný, je tu poslední možnost a tou je tzv. ostrovní mód. V tomto módu se odpojíme od části nebo od celého internetu. Ostrovní módy rozděluji do tří kategorií podle toho, s jakou částí internetu zůstane zachována komunikace.

Částečný ostrovní mód

Tento mód použijeme, pokud můžeme alespoň přibližně určit, odkud útok přichází. Zde mám na mysli skutečně geografickou polohu iniciátora útoku. Tohoto módu může být využito při obraně proti „DNS Amplification Attack“, protože většina veřejných DNS serverů podporujících rekurzi se nachází v Indii. Proto můžeme při zaznamenání tohoto útoku buď sami nebo za pomoci ISP, odpojit Indii nebo celou Asii od naší sítě a tím zastavit útok.

Výhodou je, že odpojení se od této části internetu drtivá většina českých uživatelů vůbec nepozná.

Ostrovní mód

V tomto módu zůstaneme propojeni pouze s důvěryhodnými partnery a odpojíme celý zbytek světa. Tento mód využijeme v případě, že je na nás směřován silný útok z několika jiných zemí, či světových stran. Takovouto službu poskytuje projekt FENIX sdružení NIX.CZ, který spojuje několik českých organizací, např. seznam.cz, cz.nic, CESNET a další. Tyto organizace mohou v případě rozsáhlých DoS útoků přejít do tohoto ostrovního módu a komunikovat pouze spolu navzájem, čímž zastaví útok a zachovají poměrně významnou část konektivity pro své uživatele. Běžný český uživatel by měl být útokem zasažen jen mírně a to při pokusu o přístup k zahraničním službám. Nicméně značná část českých uživatelů stále navštěvuje převážně české stránky a jejich funkčnost by neměla být zasažena. Měla by být zachována funkčnost českého zpravodajství, emailové komunikace, českých e-shopů, internetového bankovnictví apod. [7]

Úplný ostrovní mód

Tento mód použijeme, pokud je na nás veden silný útok z několika jiných zemí či světových stran, a naše společnost není součástí projektu podobného projektu FENIX. I pokud je společnost součástí podobného projektu, ale útoky začnou přicházet i ze sítí našich partnerů. V tomto módu odpojíme naši organizaci kompletně od internetu. Tento mód organizace může použít pouze, pokud to má smysl. Např. pokud naše organizace provozuje zpravodajský web, nemá pro ni úplný ostrovní mód žádný smysl. Naopak, pokud organizace provozuje a pomocí informačního sytému spravuje sklad zboží, e-shop a kamenný obchod, je funkčnost systému jako celku mnohem důležitější, než jeho připojení k internetu. Taková společnost pracující v úplném ostrovním módu může stále přijímat zboží od svých dodavatelů, vydávat zboží zákazníkům, provozovat kamenný obchod a objednávky pomocí telefonu a všechny tyto operace budou nadále uchovávány v informačním systému, nebude fungovat pouze e-shop. Pokud by firma nepřistoupila k úplnému ostrovnímu režimu, v důsledku DoS útoku by riskovala zpomalení nebo poškození celého jejího informačního systému a tím mnohem vyšší finanční ztráty. Tato možnost ochrany je ale skutečně až poslední, neboť značí výhru útočníka, který dosáhl kompletního odmítnutí naší služby prostřednictvím internetu.

Nehledě na to jak a zdali se ubráníme DoS útoku. Při každém zaznamenání DoS útoku na naši organizaci bychom měli na tuto skutečnost upozornit orgány činné v trestním řízení. Poskytneme jim logy z útoku, popřípadě IP adresu nebo identitu útočníka, pokud se nám ji podařilo zjistit.

Kapitola 4

Nástroje pro DoS útoky

Na internetu lze velmi snadno získat některý z nástrojů umožňující provést DoS útok. Konkrétně mně trvalo zhruba jedno odpoledne, než jsem našel důvěryhodný a funkční nástroj. Poté i nepříliš zkušenému uživateli stačí zadat doménové jméno serveru, popřípadě IP adresu serveru nebo uživatele, který se mu znelíbil, a nechat nástroj dělat svou práci. Tím spustí plnohodnotný DoS útok, přičemž nemusí mít žádné znalosti ohledně protokolů, na které se daný útok zaměřuje. V této kapitole popíšu několik známých a několik neznámých nástrojů pro DoS útoky.

4.1 Low Orbit Ion Cannon (LOIC)

Low Orbit Ion Cannon (česky nízko-orbitální iontový kanón), jehož jméno je převzato ze strategické hry „Command & Conquer“, je jednoduchý nástroj provádějící DoS útoky zaměřené na spotřebování šířky pásma a to generováním velkého množství TCP, UDP a HTTP provozu. LOIC byl původně vyvinut firmou Praetox Technologies jako nástroj, který měl pomoci vývojářům otestovat, zda jsou jejich servery odolné proti DoS útokům zaměřeným právě na spotřebování šířky pásma. Firma Praetox Technologies však tento nástroj zveřejnila i se zdrojovými kódy. Toho si všimla „hacktivistická“ skupina Anonymous, zdrojové kódy převzala a použila je k provedení koordinovaných DDoS útoků. Nedlouho poté Anonymous označili LOIC jako svůj oblíbený DoS nástroj a modifikovaly ho tak, aby bylo možné spouštět útoky zároveň ze všech strojů, jejichž majitelé se přihlásí k odběru IRC kanálu ovládaného Anonymous. Pomocí tohoto IRC kanálu byl administrátor schopen předávat příkazy LOIC programům běžících na strojích přihlášených k odběru. Tímto způsobem byl administrátor schopen spustit mnohem účinnější útok, než kdyby se měl spoléhal na pouhou domluvu s uživateli vlastnícími LOIC. Ke konci roku 2011 Anonymous začali upouštět od používání LOIC, a to kvůli tomu, že LOIC jako takový nijak neskrýval IP adresy útočníků. Tento nedostatek ochrany anonymity útočníků způsobil zatčení několika z těch, kteří se na DDoS útocích za použití LOIC podíleli. Nakonec se Anonymous rozhodli úplně zastavit používání LOIC. [19]

4.2 High Orbit Ion Cannon (HOIC)

High Orbit Ion Cannon (česky vysoko-orbitální iontový kanón), jehož jméno je také převzato ze strategické hry „Command & Conquer“, je podobně jako LOIC jednoduchý nástroj provádějící DoS útoky zaměřené na spotřebování šířky pásma, tentokrát však generováním

velkého množství HTTP GET a HTTP POST požadavků. Poté, co LOIC upadl v nemilost, Anonymous přešli k používání HOIC. Jedno z jeho prvních nasazení bylo při útoku na Ministerstvo spravedlnosti Spojených států amerických, který byl odplatou za zablokování provozu serveru Megaupload.com. Jeho efektivita je ještě zvýšena použitím zesilovacích skriptů napsaných v „Basicu“ a poskytnutý uživatelem v textových souborech, které během útoku může HOIC interpretovat. Ačkoliv sám HOIC nepoužívá žádnou anonymizační techniku, při použití zesilovacích skriptů může uživatel specifikovat seznam cílů a seznam identifikačních informací, které bude HOIC procházet, a za jejichž pomocí bude generovat provoz. Díky tomu nebude HOIC používat reálné informace o uživateli, čímž se stane uživatel anonymním a útok se stane hůře odvratitelným. HOIC je i v současnosti Anonymous používán při jejich DDoS útocích, ačkoliv to není jediný nástroj, který používají. [19]

4.3 Slowloris a R U Dead Yet? (R.U.D.Y.)

Slowloris (česky outloň váhavý) je pojmenována po pomalu se pohybujícím asijském primátovi. Podobně jako tento primát i Slowloris si jde za svým cílem pomalu, ale jistě. Slowloris je zástupce tzv. pomalých útoků. Je zaměřena na aplikační vrstvu, konkrétně na protokol HTTP a jeho metodu GET. Slowloris otevře desítky současných připojení k webovému serveru a v určitých intervalech odesílá části HTTP GET požadavků, které nikdy nedokončí. Server čeká na dokončení těchto požadavků a přestane přijímat nová připojení od legitimních uživatelů. Pokud je uživatel k serveru připojen před spuštěním útoku, útok na něj nemá vliv. Až se však odpojí, zabere Slowloris jeho slot připojení a znovu už ho nenechá připojit. Díky nevelkým nárokům na systém a účinnosti i proti oblíbeným webovým serverům jako např. Apache 1.x a Apache 2.x se během let Slowloris přičítá řada úspěšných útoků. Za zmínku stojí útok Íránských „hacktivistů“, kteří roku 2009 pomocí Slowloris úspěšně zaútočili na webové stránky íránské vlády. Tento nástroj byl vyvinut hackerem Robertem „RSnake“ Hansenem. Ten patří mezi hackery, kteří hackováním odhalují bezpečnostní rizika systémů proto, aby mohly být vylepšena jejich ochrana. [9]

R.U.D.Y. je pojmenované podle alba finské metalové kapely Children of Bodom. Jeho funkce je velice podobná Slowloris s tím rozdílem, že R.U.D.Y. útočí na metodu POST protokolu HTTP. Po spuštění útoku tento nástroj prochází webové stránky oběti a hledá vložený webový formulář. Po nálezu takového formuláře odešle hlavičku HTTP POST požadavku s abnormálně velkým parametrem „content-length“. Tento obsah poté posílá ve velmi malých blocích s co možná nejdelšími časovými intervaly mezi odesláním jednotlivých bloků. [8]

Výše jsem popsal známé DoS útoky. Funkčnost LOIC a HOIC jsem bohužel netestoval, neboť se mi nepodařilo sehnat jejich zdrojové kódy. Nechtěl jsem riskovat spuštění takovýchto nástrojů bez možnosti zkontrolovat, co skutečně způsobí v mém počítači. Funkčnost Slowloris a R.U.D.Y. jsem netestoval přímo s těmito nástroji, ale jejich principy jsou implementovány v nástrojích, které jsem testoval. Následující útoky jsou méně známé nebo úplně neznámé, avšak jejich autoři tyto nástroje nabízejí v otevřené formě, mohl jsem před jejich spuštěním alespoň zběžně zkontrolovat, co který nástroj skutečně činí. Soubory, o kterých budu hovořit dále v této kapitole, se nacházejí na přiloženém CD v adresáři `pcap/tools/`.

Cílem útoků je stroj „Lenovo® IdeaPad Flex 10“ s parametry:

- procesor: Intel® Pentium™ N3520 (4 x 2.42GHz)
- operační paměť: 4GB DDR3
- operační systém: Microsoft® Windows® 8 (64-bit)
- antivirus a firewall: ESET Smart Security 8
- webový server: Apache 2.2.21

4.4 GoldenEye

GoldenEye je testovací nástroj vytvořený Janem Seidlem a je zaměřený na aplikační vrstvu, konkrétně na protokol HTTP. Nástroj umožňuje provádět útok na metodu GET nebo POST. Přestože v popisu nástroje je uvedeno, že k útoku využívá „HTTP Keep Alive + NoCache“, a tudíž by se mělo jednat o útok řadící se mezi pomalé, z jeho skutečného chování při útoku můžeme usoudit, že se jedná o útok na šířku pásma, popřípadě na výkon webového serveru. Děje se tak proto, že nástroj odesílá legitimně ukončené HTTP GET a HTTP POST požadavky, takže server zpracuje odpověď a odešle ji, čímž nedochází k blokování slotu připojení. Nástroj však tyto požadavky odesílá zhruba po 5ms, čímž zatěžuje server a spotřebovává šířku pásma.

V souborech `goldeneye-get-test.pcapng` a `goldeneye-get-test.pcapng` jsou zaznamenány útoky na HTTP GET a HTTP POST spuštěné s jedním připojením v jednom vlákně. Můžeme zde vidět, že nástroj opravdu zasílá legitimní požadavky a server na ně okamžitě odpovídá. Při postupném zvyšování počtu připojení na vlákno se rychlost odpovědi serveru na požadavek snižuje. V souborech `goldeneye-get-uspesny-500.pcapng` a `goldeneye-post-uspesny-500.pcapng` jsou zaznamenány útoky na HTTP GET a HTTP POST spuštěné s pěti sty připojeními v jednom vlákně. Můžeme zde vidět, že na počátku útoku server ještě odpovídá. S postupem času se však rychlost odpovědi snižuje, až je tak nízká, že útok můžeme označit za úspěšný. Pro úspěšné provedení útoku bohužel musím vypnout firewall, protože nástroj takřka současně otevře všechna připojení, což firewall označí za „TCP SYN FLOOD“ útok.

Zhodnocení nástroje:

+ Umožňuje zvolit metodu útoku, tedy HTTP GET nebo HTTP POST.
+ Umožňuje zvolit počet vláken, které budou útočit.
+ Umožňuje zvolit počet připojení v každém vlákně.
– Neumožňuje zvolit interval odesílání útoků.
– Neumožňuje zvolit časové rozložení odesílání útoků.
– Neumožňuje zvolit časové rozložení vytváření připojení k oběti.

Tento nástroj je dostupný z [22].

4.5 ddos-toolbox

Ddos-toolbox je nástroj vytvořený Tomem Brennanem a uživatelem pod přezdívkou „sammydre“ a to pro společnost PROACTIVE RISK zabývající se internetovou bezpečností. Nástroj je zaměřen na aplikační vrstvu, konkrétně na protokol HTTP. Umožňuje volit mezi třemi druhy útoků a to „slow-headers“, „slow-post“ nebo „ssl-renegotiation“. V případě útoku „slow-headers“ lze zvolit, zda bude útok probíhat metodou GET nebo POST. Tento nástroj bohužel nemá dokumentaci, zato má vcelku obsáhlou nápovědu.

Útok typu „slow-headers“ odešle serveru úvodní část hlavičky HTTP GET nebo HTTP POST požadavku. Požadavek však nedokončí, ale v určitých časových intervalech odesílá další a další části hlavičky, čímž zabírá slot připojení. Zda budeme tento typ útoku provádět metodou GET nebo POST, nemá na funkčnost velký význam. Při testování se však ukázalo, že v případě tohoto typu útoku nástroj z nějakého důvodu packety na server neodesílá. V souborech `slow-head-get-test.pcapng` a `slow-head-post-test.pcapng` jsou zaznamenány pokusy o útok s jedním připojením jak na metodu GET, tak na metodu POST. Můžeme zde vidět, že dojde pouze k připojení k serveru, tedy provedení „TCP 3-Way Handshake“. Žádná další komunikace není zaznamenána, přestože jsou do konzole jsou vypisovány zprávy o generování správného obsahu packetů a o jejich odeslání.

Útok typu „slow-post“ odešle serveru HTTP POST požadavek s parametrem oznamujícím, že bude uživatel na server nahrávat soubor o zvolené velikosti. Obsah tohoto souboru poté posílá po jednom znaku ve zvolených časových intervalech. V tomto nástroji lze zvolit, zda bude odeslán náhodný znak nebo znak „A“ pro testovací účely. V souboru `slow-post-test.pcapng` je zaznamenán útok s jedním připojením. Můžeme zde vidět legitimní odeslání hlavičky HHTP POST požadavku a poté po jedné vteřině odesílaný znak „A“. V souboru `slow-post-uspesny-64.pcapng` je zaznamenán útok s 64mi připojeními. Děje s v něm totéž jako v předchozím případě, jen 64krát. V případě mého serveru to stačí na úspěšný útok. Nástroj obsadí všechny sloty připojení a udržuje je obsazené, čímž způsobí odmítnutí služby, protože server 65. připojení nepřijme.

Zhodnocení nástroje:

+ Umožňuje zvolit metodu útoku, tedy „slow-headers GET/POST“, „slow-post“ nebo „ssl-renegotiation“.
+ Umožňuje zvolit počet připojení.
+ Umožňuje zvolit interval odesílání útoků.
+ Umožňuje zvolit rychlost vytváření připojení k oběti.
– Neumožňuje zvolit časové rozložení odesílání útoků.
– Neumožňuje zvolit časové rozložení vytváření připojení k oběti.

Tento nástroj je dostupný z [21].

4.6 DNS Amplification Attack

Testoval jsem dva nástroje pro útoky na šířku pásma pomocí zesílení a odrazu přes DNS server. První z nich byl vytvořen Ethanem Willonerem, jako potvrzení konceptu útoků odražených od DNS serveru. U tohoto nástroje je možné nastavit pouze IP adresy oběti

a jejího portu. Proto jsem ve zdrojovém kódu upravil alespoň IP adresu DNS serveru, abych mohl nástroj otestovat. Záznam testu spuštění tohoto nástroje můžeme nalézt v souboru `dns-real-test.pcapng`. Můžeme zde vidět několik DNS dotazů na různé adresy, posílané na můj DNS server (`192.168.1.55`) a odpovědi na ně. Nástroj je spuštěn na stroji s IP adresou `192.168.1.55`, ale zdrojová IP adresa DNS požadavků je `192.168.1.25`. Zdrojová IP adresa je tedy úspěšně podvrhnuta. Není zde přiložen úspěšný útok, protože z legálních důvodů nemohu použít veřejné DNS servery a vhledem k tomu, že útoky spouštím ze stejného stroje, na kterém mi běží DNS server, nebylo by již možno hovořit o odraženém útoku a navíc by dříve došlo k zahlcení útočícího stroje, než k zahlcení oběti.

Zhodnocení nástroje:

+ Dokáže podvrhnout IP adresu odesílatele.

– Neumožňuje nastavení adres DNS serverů.

– Neumožňuje nastavení adres (URL) pro DNS dotazy.

– Neumožňuje zvolit časové rozložení odesílání útoků.

Tento nástroj je dostupný z [26].

Druhý z nich, publikovaný na portálu PacketStormSecurity.com není „DNS Amplification Attack“ v pravém slova smyslu, pouze tuto funkci simuluje. Tento nástroj totiž neposílá podvrhnuté DNS požadavky na DNS server, ale posílá podvrhnuté DNS odpovědi přímo oběti. U tohoto nástroje lze nastavit pouze IP adresa oběti. Záznam testu spuštění tohoto nástroje můžeme nalézt v souboru `dns-fake-prijate.pcapng`. Můžeme zde vidět několik DNS odpovědí s podvrhnutými a náhodně generovanými zdrojovými IP adresami a zdrojovými porty. Nepřikládám záznam úspěšného útoku, neboť pokud nechám tento nástroj chvíli pracovat, způsobí pád jak stroje oběti, tak stroje útočníka. Není proto možné tento útok zaznamenat.

Zhodnocení nástroje:

+ Způsobí pád stroje oběti.

+ Generuje náhodné zdrojové adresy.

– Způsobí pád stroje útočníka.

– Neumožňuje zvolit časové rozložení odesílání útoků.

Tento nástroj je dostupný z [2].

Kapitola 5

Návrh

V této kapitole bude popsána specifikace zadání, návrh samotného systému, návrh a popis jednotlivých mnou implementovaných útoků.

5.1 Specifikace zadání

Po základním prostudování problematiky DoS útoků a jejich typů jsme s vedoucím práce dospěli k následující specifikaci zadání. Ještě před návrhem a implementací je nutno vyhledat a otestovat dostupné DoS nástroje, poznat jejich slabé a silné stránky a tyto poznatky využít při návrhu mého DoS nástroje. Tento bod zadání jste již měli možnost prostudovat v předchozí kapitole.

Mnou vytvářený DoS nástroj (dále jen nástroj) bude implementován v programovacím jazyce C++11 a bude přeložitelný a spustitelný pod operačním systémem Linux. Funkčnost pod operačním systémem Windows je žádoucí, avšak ne bezpodmínečně nutná. Nástroj bude umožňovat provedení tří různých útoků a to útok na metodu GET a metodu POST protokolu HTTP a útok známý jako „DNS Amplification Attack“. Parametry spuštění budou nástroji předány v XML souboru, jehož strukturu mám za úkol navrhnout. Pomocí parametrů musí být umožněno nástroji zadat:

- který z útoků si přejeme provádět
- na koho má být útok směřován (IP adresa a port)
- počet připojení
- časové rozložení mezi jednotlivými packety a to alespoň:
  - konstantní
  - exponenciální
- velikost obsahu (POST útok)
- datové rozložení jednotlivých packetů (POST útok) a to alespoň:
  - konstantní
  - exponenciální
- IP adresy DNS serverů (DNS útok)

Nástroj by měl umožňovat spustit jednotlivé útoky, spouštět kombinace dvou útoků, popřípadě spustit všechny tři útoky zároveň. Po dohodě s vedoucím jsme upravili druhý bod původního zadání a to vypuštěním použití knihovny Pcap, a to z důvodu, že při použití tzv. „Raw socketů“ není její použití potřeba.

5.2 Parametry programu

Jak již bylo zmíněno, nástroji budou parametry předány pomocí XML souboru. V tomto souboru se bude vždy nacházet kořenová značka `<dos>`. Uvnitř této značky se musí nacházet jedna nebo více značek `<attack>` pro definici jednotlivých útoků. Uvnitř této značky bude jedna ze značek `<get>`, `<post>` nebo `<dns>`, které budou obsahovat konkrétní parametry pro daný útok.

Pro všechny implementované útoky je možné nastavit zadáním požadované parametry, tedy IP adresu a port oběti, počet připojení a časové rozložení mezi odesláním packetů pro jednotlivá připojení. Navíc je oproti zadání možno nastavit časové rozložení odloženého startu a to proto, aby se nenavázala všechna připojení naráz, což může způsobit zablokování komunikace firewallem. Tento parametr je volitelný. U všech zadávaných rozložení je možno volit mezi zadáním požadovanými rozloženími, tedy konstantním a exponenciálním a navíc můžeme volit z rozložení uniformního a Gaussova (normálního). U všech zadávaných časových údajů je tento údaj zadáván v milisekundách. U rozložení zadáváme údaje pro dané rozložení typické. U exponenciálního a Gaussova rozložení můžeme navíc volitelně zadat minimum a maximum.

V případě GET útoku je to všechno, co potřebujeme. U POST útoku je ještě třeba zadat velikost posílaného souboru. Tato hodnota se nastavuje v Bytech (počtu znaků). Dále je třeba zadat datové rozložení, tedy po jak velkých blocích se budou data serveru posílat. Tato hodnota se stejně jako velikost souboru zadává v Bytech (počtu znaků). V případě DNS útoku zadáváme jednu či více IP adres DNS serverů, na které budeme posílat dotazy. Systém otevře jedno připojení pro každý DNS server, proto se u DNS útoku explicitně nezadává počet připojení. Dále zde zadáváme URL adresu, jejíž rezoluci budeme od serverů požadovat.

Pro přesné názvy a hierarchii značek prosím nahlédněte do souboru `params/dos.dtd`, který obsahuje DTD definici XML souboru sestavenou podle výše uvedeného návrhu.[5] DTD soubor ovšem není příliš názorný, proto v souboru `params/get_post_dns.xml` můžeme nalézt ukázku nastavení parametrů pro všechny útoky současně a se všemi možnými rozloženími.

5.3 Jádro systému

Jelikož je zadáním požadováno dodržovat zvolené časové rozložení při odesílání packetů, není možné je pouze v cyklu vytvářet a odesílat. V mém nástroji bude požadovaného chování docíleno tím, že útoky budou ovládány simulačním jádrem kombinované simulace, kde diskrétní částí simulace bude provedení jednoho kroku útoku, neboli provedení jedné události z kalendáře událostí. Za spojitou část simulace můžeme považovat čekání do aktivačního času události. Simulační jádro bude provádět algoritmus „next-event“ nad předaným kalendářem událostí. Vytáhne první záznam z kalendáře událostí, počká do aktivačního času a poté tuto událost provede. Tento úkon bude prováděn tak dlouho, dokud nedojde k vyprázdnění kalendáře událostí. V kalendáři událostí budou uchovávány záznamy událostí,

které mají být provedeny a jejich aktivační časy. Budou podle těchto časů seřazeny.

Protože chceme, aby bylo možné provádět všechny útoky současně a nezávisle na sobě, bude každý z útoků spuštěn v jednom vlákně. Každý útok bude reprezentován třídou. V těchto třídách a jejich statických proměnných budou uchovávána obecná nastavení útoku. Každé připojení bude reprezentováno instancí dané třídy a instanční proměnné budou sloužit k uchovávání stavu daného připojení. Statické metody budou sloužit k nastavení parametrů a ovládání útoku. Nestatické metody budou sloužit k samotnému provádění útoku. Každý útok (třída) bude mít svůj vlastní kalendář událostí. Tento kalendář události bude předán ke zpracování simulačnímu jádru. Na konci každé z prováděných událostí (jednoho kroku útoku) bude zapotřebí naplánovat do kalendáře událostí spuštění další události (dalšího kroku útoku) a to na požadovaný čas.

5.4 GET útok

Mnou implementovaný GET útok se řadí do skupiny tzv. pomalých útoků. Je tedy zaměřen na obsazení všech slotů připojení serveru a udržování těchto slotů obsazených. Díky tomu, že HTTP GET požadavek nemusí být odeslán v jednom packetu a servery počítají s tím, že někteří uživatelé mají pomalé připojení k internetu (např. vytáčené). Na serverech jsou nastavené poměrně dlouhé čekací doby na další část HTTP GET požadavku. Tohoto chování využijeme k útoku a to tak, že odešleme legitimní část HTTP GET hlavičky na server, ale neodešleme ukončovací sekvenci „`\r\n`“. Server bude čekat na další část požadavku. Po uplynutí požadované doby pošleme serveru další část HTTP GET hlavičky, nyní již s náhodně generovaným obsahem a opět nepřipojíme ukončovací sekvenci. Takto udržujeme spojení se serverem otevřené co nejdéle. Při otevření dostatečného množství spojení již nebude server moci přijmout spojení od legitimních uživatelů.

GET útok si tedy vytvoří kalendář událostí pro provádění jednotlivých fází útoku a naplní ho událostmi pro zahájení útoku. Po naplnění kalendáře ho předá simulačnímu jádru, které bude události postupně provádět. Nyní popíšu, jaké události bude GET útok obsahovat a jaký bude jejich účel. V zahajovací události otevřeme socket, připojíme se k oběti a naplánujeme odeslání úvodní části hlavičky. V této události, jak již napovídá název, dojde k vytvoření a odeslání úvodní legitimní části hlavičky HTTP GET požadavku a naplánuje se odeslání další části hlavičky. Tato událost náhodně vygeneruje a odešle serveru další část hlavičky HTTP GET požadavku a naplánuje své další spuštění. Vzhledem k tomu, že implementuji útok, je třeba reagovat na některé chyby. Pokud na některou z odeslaných částí hlavičky server odpoví chybovým kódem, znovu naplánujeme odeslání úvodní části hlavičky. Pokud dojde k uzavření spojení, naplánujeme zahajovací událost útoku. Při plánování událostí se náhodně generují časy spuštění podle požadovaného rozložení.

5.5 POST útok

Podobně jako GET útok se i mnou implementovaný POST útok řadí do skupiny tzv. pomalých útoků. Také je zaměřen na obsazení všech slotů připojení serveru a udržování těchto slotů obsazených, nicméně mírně odlišným způsobem, než je použit u GET útoku. V případě POST útoku serveru odešleme legitimní hlavičku, v níž mu parametrem „Content-Length“ oznámíme, že budeme odesílat soubor o zvolené velikosti. Podobně jako u metody GET i u metody POST servery počítají s obsluhou uživatelů s pomalým připojením k internetu a proto jsou i zde čekací doby na další část souboru poměrně dlouhé. Proto serveru části

souboru o zvolené velikosti a náhodným obsahem odesíláme ve zvolených intervalech. Po odeslání celého souboru proces opakujeme. Takto udržujeme spojení se serverem otevřené, jak dlouho si přejeme. Pokud takovýchto spojení otevřeme dostatek, server již nebude moci přijmout spojení od legitimních uživatelů. Navíc je zde možnost, že odesílání velkého počtu velkých souborů spotřebuje úložný prostor nebo RAM paměť serveru, což může vést k nestandartnímu chování.

POST útok si také připraví kalendář událostí pro provádění jednotlivých fází útoku a naplní ho událostmi pro zahájení útoku. Po naplnění kalendáře ho předá simulačnímu jádru, které bude události postupně provádět. Nyní popíšu, jaké události bude POST útok obsahovat a jaký bude jejich účel. V zahajovací události otevřeme socket, připojíme se k oběti a naplánujeme odeslání hlavičky. V této události se vytvoří a odešle HTTP POST hlavička, v níž serveru oznámíme, že mu budeme odesílat soubor o zvolené velikosti a naplánuje se odeslání části souboru. Tato událost vygeneruje náhodný obsah souboru o zvolené nebo náhodné velikosti a odešle ho serveru. Dále naplánuje své další spuštění. Pokud zbývá k odeslání méně znaků, než by si událost pro odeslání části souboru přála, přejdeme do události odeslání „patičky“. V této události se náhodně vygeneruje a odešle zbytek souboru a naplánuje se opětovné odeslání hlavičky. Podobně jako u GET útoku je třeba i zde reagovat na některé chyby. Pokud na některou z odeslaných částí souboru server odpoví chybovým kódem, znovu naplánujeme odeslání hlavičky. Pokud dojde k uzavření spojení, naplánujeme zahajovací událost útoku. Při plánování událostí se náhodně generují časy spuštění podle požadovaného rozložení.

5.6 DNS útok

Mnou implementovaný DNS útok se řadí mezi útoky na šířku pásma, útoky odražené a zesílené. U tohoto útoku se posílají DNS požadavky s podvrhnutou IP adresou odesílatele na DNS server. DNS server provede rezoluci a DNS odpověď pošle oběti. Pokud je odesláno dostatečné množství DNS požadavků na dostatečné množství DNS serverů, dojde k vyčerpání šířky pásma oběti.

DNS útok si také připraví kalendář událostí pro provádění jednotlivých fází útoku a naplní ho událostmi pro zahájení útoku. Po naplnění kalendáře ho předá simulačnímu jádru, které bude události postupně provádět. V zahajovací události se otevře „RAW socket“ a naplánuje se odeslání DNS požadavku. V této události se vytvoří DNS požadavek, vytvoří a vyplní se UDP hlavička, vytvoří a vyplní se IP hlavička. Vytvořený datagram se odešle DNS serveru a naplánuje se odeslání DNS požadavku. Při plánování událostí se náhodně generují časy spuštění podle požadovaného rozložení.

Kapitola 6

Implementace

V předchozí kapitole, tedy návrhu jsem, poněkud abstraktně popsal strukturu, funkčnost a logiku jednotlivých útoků. V této kapitole se podíváme více do hloubky. Soubory, na které budu odkazovat v dalším textu, se nalézají v adresáři `source/`.

Nejdříve popíši to, co mají všechny útoky společné. Jak jsem se již výše zmínil, každý útok je reprezentován jednou třídou. Deklarace těchto tříd můžeme nalézt v souborech `GETattack.hpp`, `POSTattack.hpp` a `DNSattack.hpp`. Každá z těchto tříd ve svých statických proměnných uchovává všechny parametry, které se dají útokům nastavit. Pro nastavení hodnot těchto proměnných jsou implementovány settery. Třídy ve statických proměnných uchovávají rovněž dva kalendáře událostí a generátor pseudonáhodných čísel. Jeden z kalendářů událostí, `calendar_of_events`, slouží pro standartní běh útoku. Druhý z nich, `calendar_of_end_events`, uchovává ukončovací metody útoků a používá se k ukončení útoku. Každý z útoků disponuje statickou metodou `start()`, která vytvoří tolik instancí sama sebe, kolik je požadováno připojení v daném útoku. Pro každou z instancí vytvoří dva aktivační záznamy. Do jednoho z nich vloží wrapper startovací metody útoku, tedy `sockOpen()`.[12] Do druhého z nich vloží wrapper ukončovací metody útoku, tedy `sockClose()`. Aktivační čas obou událostí nastaví na nulu, aby byly vykonány okamžitě. První aktivační záznam vloží do kalendáře `calendar_of_events` a druhý záznam vloží do kalendáře `calendar_of_end_events`. Následně předá první kalendář událostí (`calendar_of_events`) metodě `run()` třídy `AttackConrol`, která začne invokovat metody v kalendáři uložené, čímž se spustí provádění útoku. Po navrácení řízení se metodě `run()` třídy `AttackConrol` předá druhý kalendář událostí (`calendar_of_end_events`), čímž se spustí korektní ukončování útoku. Třídy disponují statickou metodou `end()`, která deaktivuje kalendář událostí pro běh `calendar_of_events` invokací jeho metody `deactivate()`. Třídy útoků dědí od třídy `enable_shared_from_this` a to proto, že v celém programu využívám smart pointerů importovaných ze standartní knihovny `memory`, které fungují podobně jako garbage collector v některých jiných programovacích jazycích.[13]

Všechny třídy útoků také obsahují metody podpůrné, které přímo nesouvisí s prováděním útoku, ale jsou důležité pro jeho správný chod. Nejdůležitější z nich je nestatická metoda `planMe()`, která přijímá wrapper metody, jejíž spuštění má být naplánováno. Vytvoří aktivační záznam této metody a naplánuje její spuštění, tedy vloží aktivační záznam do kalendáře událostí. Další důležitou metodou je nestatické metoda `getPtr()`, která pomocí zděděné metody `shared_from_this()` vrací `shared_ptr`, což je ukazatel na instanci, ve které se právě nacházíme, takže něco jak `this`, jen že za použití smart pointerů.[13] Tato metoda se využívá při vytváření wrapperů metod pro vytvoření aktivačního záznamu události. Dále jsou zde statické metody pro vygenerování náhodného čísla `getRandom()`,

které zapouzdřují práci s generátorem pseudonáhodných čísel. Ta důležitější z nich jako parametr přijímá požadovaný typ náhodného čísla. Chceme-li tedy generovat počáteční zpoždění `START_DELAY`, nebo zpoždění jednotlivých packetů během útoku `PACKET_DELAY`. Tato metoda automaticky generuje čísla s uživatelem požadovaným rozložením. Statická metoda `getActTime()` pomocí metody `getRandom()` získá náhodné číslo a to přičte k aktuálnímu času, čímž získá aktivační čas události. Tento čas vrací ve struktuře `timeval`. Další podpůrnou funkcí je funkce `getRandStr()`, která generuje a vrací řetězec náhodných znaků a jako parametr přijímá počet generovaných znaků. Tato funkce je implementována v souboru `main.cpp`.

6.1 Parametry programu

Parser parametrů je volán z funkce `main()` a jeho deklarace se nachází v souboru `ParameterParser.hpp`. Pro parsování parametrů používám knihovnu `libxml2` dostupnou z [25]. Jako první se pomocí funkce `xmlCtxtReadFile()` provede DTD validace XML souboru a sestaví se XML strom. Úroveň attack XML stromu se v cyklu prochází a pro každý uzel attack se dále zanořuje. Zde dojde k zjištění, který z útoků je definován, invokuje se příslušná metoda (`parseGET()`, `parsePOST()` nebo `parseDNS()`) a v ní dochází k procházení této větve XML stromu. V těchto metodách se v procházené větvi XML stromu zanořujeme až k listům.Jejich hodnoty jsou pomocí setterů předány a nastaveny do statických proměnných příslušné třídy útoku. V těchto metodách se také kontroluje správnost zadaných IP adres a portů a to pomocí funkcí `gethostbyname()` a `getservbyport()`. Pokud je IP adresa nebo port nesprávný, končí metoda a dále celý program chybou. IP adresy jsou ukládány v datovém typu `string` a to kvůli kompatibilitě mezi systémy Linux a Windows. Pokud se během procházení této větve narazí na definici rozložení, ať už časového nebo datového, předá se tato část větve metodě `parseDistr()`. Metoda projde tuto podčást stromu, rozpozná, o jaké rozložení se jedná, a nastaví příslušné proměnné hodnotami v listech. Po ukončení této metody se hodnoty opět pomocí setteru předají příslušné třídě. Ve většině případů probíhá procházení přímočaře, pouze u parsování DNS útoku se uzly `dns_ip` procházejí v cyklu, protože jich může být zadáno 1 až N.

6.2 Jádro systému

Základní tok programu je stejný pro všechny útoky. Po nastavení parametrů útoků se do seznamu `start_vec` uloží startovací (`start()`) metody zvolených útoků. Do seznamu `end_vec` se uloží ukončovací (`end()`) metody zvolených útoků. Toto uložení se provádí pomocí wrapperů metod vytvořených pomocí standardní knihovny `functional` a funkce `bind()`.[12] Ve for cyklu se pro každý definovaný útok otevře nové vlákno, ve kterém se spustí obalovací funkce `startMethodWrapper()`, které se předá aktuální index z for cyklu. V této funkci se za pomocí indexu invokuje startovací metoda jednoho z útoků uložená v seznamu `start_vec`. Funkce metody `start()` je popsána v předchozím odstavci. Útoky se nyní provádí nezávisle každý ve svém vláknu. Poté, co se ve funkci `main()` vytvoří všechna vlákna, hlavní vlákno čeká na dokončení útoků, ukončení a spojení synovských vláken. Program je korektně ukončen přijetím a zpracováním některého ze sledovaných signálů, např. `SIGSTP` (ctrl + z). Zpracování signálů provádí funkce `signalHandler()`, která invokuje všechny ukončovací metody ze seznamu `end_vec` a všem otevřeným vláknům pošle signál `SIGUSR1`. Zpracování tohoto signálu probíhá ve stejné funkci jako zpracování signálů ostatních, nicméně se při

něm nic neprovádí. Vláknům je signál `SIGUSR1` odesílán kvůli tomu, aby byla probuzena v případě, že jsou zrovna uspána funkcí `nanosleep()`.

Toto byl tok programu z pohledu hlavního vlákna. Samotné útoky jsou, jak již bylo zmíněno, ovládány simulačním jádrem kombinované simulace. Toto řízení je reprezentováno statickou třídou `AttackControl` deklarovanou v souboru `AttackControl.hpp`. Hlavní částí řízení útoku je metoda `run()`, která je volána z metody `start()` jednotlivých tříd útoků. Tato metoda jako parametr přijímá ukazatel na kalendář událostí. Pokud není předaný kalendář událostí prázdný, vytáhne z něj metoda první aktivační záznam, a to zavoláním metody `popFront()` třídy `CalendarOfEvents`. Jeho aktivační čas předá své metodě `goSleep()` a po navrácení řízení invokuje metodu z aktivačního záznamu a cyklus znovu opakuje. Metoda `goSleep()` zjistí aktuální čas, od aktivačního času události aktuální čas odečte a uspí se na získanou dobu. Pokud vyjde záporný výsledek, tedy aktuální čas je větší než aktivační čas, neuspává se a pokračuje rovnou dál. Jak si můžeme všimnout, třída `AttackControl` se nijak nestará o ukončení útoku, jednoduše cyklí nad kalendářem událostí, dokud obsahuje nějaké aktivační záznamy.

Jak vyplývá z předchozího odstavce, simulační jádro pro svůj chod používá kalendář událostí. Tento kalendář událostí je deklarován v souboru `CalendarOfEvents.hpp`. Zde jsou deklarovány dvě třídy. První třídou `ActivationReccord` je reprezentovaná datová struktura uchovávající jednotlivé aktivační záznamy. Uchovává tedy informaci o aktivačním času záznamu a wrapper metody, která má být v daný čas invokována. Druhou třídou je samotný kalendář událostí `CalendarOfEvents`, který uchovává seřazený seznam aktivačních událostí, tedy ukazatelů na instance třídy `ActivationRecord`. Tento seznam je v jazyce c++ reprezentovaný datovou strukturou `vector` a je seřazený vzestupně podle aktivačního času události. Metoda `add()` přijímá ukazatel na instanci `ActivationRecord` a tento aktivační záznam zařadí do seřazeného seznamu na místo, kam podle aktivačního času události patří. Před vložením aktivačního záznamu události do kalendáře si kalendář zkontroluje, zda je aktivní. Tímto způsobem je řešeno korektní ukončení útoku. Invokací metody `deactivate()` dojde k deaktivování kalendáře událostí a vymazání jeho obsahu. Jelikož je po deaktivaci kalendář prázdný, a není možné do něj vložit další aktivační záznam, cyklus v metodě `run()` třídy `AttackControl` se ukončí. Další důležitou metodou třídy `CalendarOfEvents` je metoda `popFront()`, která získá první aktivační záznam z kalendáře, smaže ho z kalendáře a ukazatel na něj vrátí.

6.3 GET útok

Ovládací část funkce GET útoku byla popsána v úvodu této kapitoly, nyní popíši samotný útok. Útok začíná invokací metody `sockOpen()`. V této metodě dojde k vytvoření socketu pro TCP spojení. Nastaví se, aby udržoval spojení otevřené. Rovněž se nastaví `SO_RCVTIMEO`, tedy časový limit pro přijetí odpovědi od serveru a naplánuje se spuštění metody `victimConnect()` s počátečním zpožděním (`START_DELAY`). Metoda `victimConnect()` se pomocí funkce `connect()` připojí k oběti. Pokud se připojí úspěšně, naplánuje spuštění metody `sendFrstHeaderPart()`. Pokud se úspěšně nepřipojí, naplánuje spuštění sebe sama.

Metoda `sendFrstHeaderPart()` volá metodu `getFrstHeaderPart()`, která vrací první část HTTP GET hlavičky. Po získání této hlavičky metoda pomocí funkce `write()` odešle hlavičku oběti. Pokud nastane chyba socketu, naplánuje se spuštění metody `sockOpen()`. Jestliže nastane chyba při odesílání, naplánuje se spuštění metody `victimConnect()`. Když vše proběhne v pořádku, naplánuje se spuštění metody `sendNextHeaderPart()`. V metodě `getFrstHeaderPart()` probíha sestavení HTTP GET hlavičky. Obsah hlavičky je sestaven

podle znalostí získaných v předmětu Tvora webových stránek (ITW), odchycených packetů posílaných webovým prohlížečem a příslušného RFC2616. [10]

Metoda `sendNextHeaderPart()` volá metodu `getNextHeaderPart()`, jež vrací náhodný počet náhodných znaků, který prohlásíme, za pokračování HTTP GET hlavičky. Následně jsou tato data opět pomocí funkce `write()` odeslána oběti. Pokud dojde k chybě socketu, naplánuje se spuštění metody `sockOpen()`. Jestliže nastane chyba při odesílání, naplánuje se spuštění metody `victimConnect()`. Nenastane-li ani jedna ze zmíněných chyb, pomocí funkce `read()` obdržíme odpověď serveru. Neobdržíme-li odpověď ve zvoleném časovém intervalu, naplánuje se znovu spuštění metody `sendNextHeaderPart()`. V opačném případě odpověď serveru předáme metodě `checkForError()`. Tato metoda zjistí, zda server odpověděl chybovým kódem, tedy 4xx nebo 5xx. Pakliže server odpověděl chybovým kódem, naplánuje se spuštění metody `sendFrstHeaderPart()`. Jestli neodpověděl chybovým kódem, naplánuje se spuštění metody `sendNextHeaderPart()`.

Tento běh se neustále opakuje, neboť v každé metodě dojde k naplánování spuštění další metody. Pokud systém zachytí signál a dojde k přerušení tohoto cyklu útočení je invokována metoda `sockClose()`, která uzavře socket a skončí. Následně skončí metoda `start()` a řízení se navrátí do funkce `startMethodWrapper()`, v níž se zavoláním funkce `pthread_exit()` ukončí toto vlákno.

6.4 POST útok

Ovládací část funkce POST útoku byla popsána v úvodu této kapitoly, nyní popíši samotný útok. Útok začíná invokací metody `sockOpen()`. V této metodě dojde k vytvoření socketu pro TCP spojení. Nastaví se, aby udržoval spojení otevřené. Rovněž se nastaví `SO_RCVTIMEO`, tedy časový limit pro přijetí odpovědi od serveru a naplánuje se spuštění metody `victimConnect()` s počátečním zpožděním (`START_DELAY`). Metoda `victimConnect()` se pomocí funkce `connect()` připojí k oběti. Pokud se připojí úspěšně, naplánuje spuštění metody `sendHeader()`, nepřipojí-li se úspěšně, naplánuje spuštění sebe sama.

Metoda `sendHeader()` volá metodu `getHeader()`, která vrací HTTP POST hlavičku. Podle požadavku uživatele se nastaví zbývající počet znaků (`content_rest`) k odeslání. Po získání HTTP POST hlavičky metoda pomocí funkce `write()` odešle hlavičku oběti. Pokud nastane chyba socketu, naplánuje se spuštění metody `sockOpen()`. Jestliže nastane chyba při odesílání, naplánuje se spuštění metody `victimConnect()`. Na závěr, proběhne-li vše proběhne v pořádku, naplánuje se spuštění metody `sendPartOfData()`. V metodě `getHeader()` probíha sestavení HTTP POST hlavičky. V ní je pomocí parametrů zakázáno kešování. Je nastaveno tak, aby se spojení udržovalo otevřené, je nastaven formát odesílaných dat a délka odesílaných dat. Obsah hlavičky byl sestaven podle znalostí získaných v předmětu Tvora webových stránek (ITW), odchycených packetů testovaných nástrojů a příslušného RFC2616. [10] [21] [22]

Metoda `sendPartOfData()` volá metodu `getPartOfData()`, která vrací podle požadovaného rozložení vygenerovaný náhodný počet náhodných znaků. Pokud metoda `getPartOfData()` vygenerovala požadovaná data, jsou tato data opět pomocí funkce `write()` odeslána oběti. Jestliže vrátila prázdný řetězec, voláme metodu `sendFooter()`. V případě, že nastane chyba socketu, naplánuje se spuštění metody `sockOpen()`. Nastane-li chyba při odesílání, naplánuje se spuštění metody `victimConnect()`. Když nenastane ani jedna ze zmíněných chyb, pomocí funkce `read()` obdržíme odpověď serveru. Pokud odpověď neobdržíme ve zvoleném časovém intervalu, naplánuje se znovu spuštění metody `sendPartOfData()`. V opačném případě odpověď serveru předáme metodě `checkForError()`. Tato metoda zjistí, zda

server odpověděl chybovým kódem, tedy 4xx nebo 5xx. Nastala-li tato eventualita, naplánuje se spuštění metody `sendHeader()`. Pakliže neodpověděl chybovým kódem, naplánuje se spuštění metody `sendPartOfData()`. Metoda `getPartOfData()` si vygeneruje náhodné číslo (`part_size`) podle zvoleného rozložení. Pokud je toto číslo větší než počet znaků zbývajících k odeslání, vrací prázdný řetězec. Je-li číslo v pořádku, zbývající počet znaků (`content_rest`) se o toto číslo dekrementuje. Následně se vygeneruje řetězec náhodných znaků o délce `part_size` a tento řetězec se vrací.

Metoda `sendFooter()` volá metodu `getFooter()`, která vrací řetězec náhodných znaků o délce počtu znaků zbývajících k odeslání (`content_rest`). Tato data jsou opět pomocí funkce `write()` odeslána oběti. Pokud nastane chyba socketu, naplánuje se spuštění metody `sockOpen()`. Nastane-li chyba při odesílání, naplánuje se spuštění metody `victimConnect()`. Jelikož jsme odeslali celý obsah, naplánuje se spuštění metody `sendHeader()`.

Tento běh se neustále opakuje, protože v každé metodě dojde k naplánování spuštění další metody. Pokud systém zachytí signál a dojde k přerušení tohoto cyklu útočení, je invokována metoda `sockClose()`, která uzavře socket a skončí. Následně skončí metoda `start()` a řízení se navrátí do funkce `startMethodWrapper()`, v níž se zavoláním funkce `pthread_exit()` ukončí toto vlákno.

6.5 DNS útok

Ovládací část funkce DNS útoku byla popsána v úvodu této kapitoly, nyní popíši samotný útok. Útok začíná invokací metody `sockOpen()`. V této metodě dojde k vytvoření RAW socketu a to pomocí přepínačů `SOCK_RAW` a `IPPROTO_RAW`. Dále se naplánuje spuštění metody `sendDNSmessage()` s počátečním zpožděním (`START_DELAY`).

Metoda `sendDNSmessage()` volá metodu `getDatagram()`, které předáváme souvislý blok paměti o velikosti `DATAGRAM_SIZE`. Tato metoda v paměti vytvoří a vrátí připravený IP datagram obsahující UDP datagram a DNS požadavek. Po získání tohoto datagramu jej metoda pomocí funkce `sendto()` odešle DNS serveru. Nastane-li chyba socketu, naplánuje se spuštění metody `sockOpen()`, jinak se znovu plánuje spuštění metody `sendDNSmessage()`.

Metoda `getDatagram()` volá metodu `getDNSmessage()`, která vrací připravený DNS požadavek. Na začátku předaného bloku paměti se vytvoří IP hlavička a za ní UDP hlavička. Struktury IP a UDP hlaviček jsou definovány v hlavičkových souborech `netinet/ip.h` a `netinet/udp.h`. Na obrázku 6.1 můžeme vidět strukturu IP hlavičky.[18] Za tyto dvě hlavičky nakopírujeme získaná data, tedy DNS požadavek. Poté se vytvoří UDP pseudo-hlavička, jejíž strukturu jsem nadefinoval v souboru `DoSlib.hpp`. Na obrázku 6.2 vidíme strukturu UDP hlavičky (vlevo) a UDP psedo-hlavičky (vpravo).[17] Vyplní se UDP hlavička a poté se vyplní UDP pseudo-hlavička. Následně se připraví souvislý blok paměti pro UDP pseudo-datagram. Ten potřebujeme pro spočítání kontrolního součtu. Na začátek UDP pseudo-datagramu se zkopíruje UDP pseudo-hlavička, za ní UDP hlavička a data (náš DNS požadavek). Voláním funkce `checksum()` převzaté z [16], které je předán UDP pseudo-datagram, se spočítá jeho kontrolní součet a ten je vložen do UDP hlavičky. Poté se vyplní IP hlavička a stejnou funkcí se spočítá kontrolní součet IP datagramu a výsledek se uloží do IP hlavičky. Tím je IP datagram dokončen a vrací se jeho délka. [16] [17] [18]

Metoda `getDNSmessage()` vytvoří DNS hlavičku definovanou v `DoSlib.hpp`. Tuto DNS hlavičku můžeme vidět na obrázku 6.3. DNS hlavička se vyplní požadovanými hodnotami. Klíčová je zde značka vyžadující po DNS serveru provedení rekurzivního překladu. Následně se zavolá metoda `getDNSaddress()`, která převede adresu z tvaru URL do tvaru přijíma-

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Obrázek 6.1: IP hlavička [18]

```
 0      7 8     15 16    23 24    31   0      7 8     15 16    23 24    31
+--------+--------+--------+--------+ +--------+--------+--------+--------+
|     Source      |   Destination   | |          source address           |
|      Port       |      Port       | +--------+--------+--------+--------+
+--------+--------+--------+--------+ |        destination address        |
|                 |                 | +--------+--------+--------+--------+
|     Length      |    Checksum     | |  zero  |protocol|   UDP length   |
+--------+--------+--------+--------+ +--------+--------+--------+--------+
|
|          data octets ...
+---------------- ...
```

Obrázek 6.2: UDP hlavička (vlevo) a UDP pseudo-hlavička (vpravo) [17]

ného DNS službou. Tato nová adresa se nakopíruje za DNS hlavičku a za touto adresou se vytvoří DNS patička, která je rovněž definovaná v souboru DoSlib.hpp. Tuto patičku tvoří ve skutečnosti dvě poslední pole DNS dotazu, jenž můžeme vidět na obrázku 6.4. Po vyplnění patičky hodnotami je DNS dotaz hotový a vracíme jeho délku. Při vytváření DNS dotazu jsem postupoval podle odchycených packetů vygenerovaných nástrojem `nslookup` a také podle příslušného RFC1035. [15]

Metoda `getDNSaddress()` převádí adresu URL na formát přijímaný DNS službou a to tak, že namísto teček a před začátek adresy je vložen počet následujících znaků, např. URL www.google.com přeloží na `[0x03]www[0x06]google[0x03]com[0x00]`. [15]

Tento běh se neustále opakuje, stejně jako v případě GET nebo POST útoku.

```
  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      ID                       |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|QR|   Opcode  |AA|TC|RD|RA|   Z    |   RCODE   |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    QDCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ANCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    NSCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ARCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Obrázek 6.3: DNS hlavička [15]

```
  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                                               |
/                     QNAME                     /
/                                               /
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     QTYPE                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     QCLASS                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Obrázek 6.4: DNS dotaz [15]

Všechny používané konstanty jsou definovány v souboru `DoSlib.hpp`. Jsou zde definovány např. typy rozložení, konstanty používané při tvorbě DNS požadavku v průběhu DNS útoku a také výchozí hodnoty používané u všech setterů pro nastavení parametrů jednotlivých útoků. V tomto souboru jsou také importovány všechny používané knihovny. Nástroj jsem se pokusil implementovat i pro chod pod systémem Windows, bohužel se mi dlouhou dobu nedařilo zprovoznit knihovny třetích stran. Po zprovoznění již nezbývalo mnoho času do odevzdání, proto jsem tuto funkcionalitu nakonec nedoimplementoval. V současné době je program pod systémem Windows přeložitelný a spustitelný, nicméně nemá žádnou funkcionalitu. Zdrojový kód je však připraven na doimplementování podpory systému Windows. Všechny funkce, které se v systému Windows liší od funkcí systému Linux, jsou obaleny v makrech pro překladač a bylo by potřeba doimplementovat jejich funkcionalitu.

Kapitola 7

Testování

Po dokončení implementace nástroje jsem jeho funkčnost otestoval podobně jako funkčnost nástrojů testovaných v kapitole 4. Soubory se záznamy útoků se nacházejí na přiloženém CD v adresáři `pcap/` a soubory s parametry útoků nacházejí na přiloženém CD v adresáři `params/`. Útoky byly směřovány na stroj „Lenovo® IdeaPad Flex 10“ s parametry:

- procesor: Intel® Pentium™ N3520 (4 x 2.42GHz)
- operační paměť: 4GB DDR3
- operační systém: Microsoft® Windows® 8 (64-bit)
- antivirus a firewall: ESET Smart Security 8
- webový server: Apache 2.2.21

Útočení bylo prováděno z virtuálního stroje běžícího v programu VirtualBox 5.0.16 s parametry:

- procesor: Intel® Core™ i5-750 (2 x 2.66GHz)
- operační paměť: 1GB DDR3
- operační systém: Ubuntu 14.04 (32-bit)
- překladač: g++ 4.8.4

7.1 GET útok

Nejdříve jsem testoval funkci GET útoku. Testovací útok jsem spouštěl s parametry, které jsou k nalezení v souboru `GET-test.xml`. Nejdůležitějším parametrem je počet připojení. Tento útok spouštím pouze s jedním připojením, aby bylo dobře viditelné, co se během něj děje. V souboru `GET-test.pcapng` nebo příloze C.1 můžeme vidět záznam testování tohoto útoku. Můžeme si všimnout, že v packetu číslo 4, který je také zobrazen v příloze C.1, dochází k odeslání první části HTTP GET hlavičky. Následně v packetech 6 a 8 jsou odesílána náhodná data. V 10. packetu server (oběť) odpoví chybou 400. Nástroj to rozpozná a ve 14. packetu opět můžeme vidět první část HTTP GET hlavičky. Poté pokračuje v posílání náhodných dat (packety 16 a 18). Z toho vyplývá, že pokud server odpoví chybou, nástroj znovu posílá úvodní část HTTP GET hlavičky. Udržuje tak slot připojení stále obsazený.

Pro úspěšný útok je třeba zvyšovat počet připojení tak dlouho, dokud nedosáhneme maximálního počtu současných připojení k serveru. V případě mého serveru je to 64 současných připojení. Úspěšný GET útok jsem spouštěl s parametry ze souboru `GET-uspesny.xml`, ve kterém otvírám právě 64 připojení. Záznam tohoto útoku se nachází v souboru `GET-uspesny.pcapng`.

7.2 POST útok

Dále jsem provedl test funkce POST útoku. Testovací útok jsem spouštěl s parametry, které jsou k nalezení v souboru `POST-test.xml`. V tomto testu se otevře jedno vlákno a v něm se bude posílat obsah o velikosti 255bytů po částech velkých 64bytů. Tento útok spouštím pouze s jedním připojením, aby bylo dobře viditelné, co se během něj děje. V souboru `POST-test.pcapng` nebo příloze C.2, můžeme vidět záznam testování tohoto útoku. V packetu číslo 4, který je také zobrazen v příloze C.2, můžeme vidět odeslání HTTP POST hlavičky, ve které je mimo jiné nastavena velikost odesílaného obsahu. V packetech 6, 8, 10 je odesílán náhodně generovaný obsah o velikosti 64bytů, v packetu číslo 12 je odesláno zbylých 63bytů. Následuje automatická komunikace nástroje se serverem. V packetu číslo 65 je opět poslána HTTP POST hlavička a v packetech 67, 69, 71 je opět odesílán náhodný obsah. Můžeme tedy vidět, že po odeslání souboru nástroj se serverem chvíli provádí automatickou komunikaci, a poté nástroj odesílá další soubor, čímž udržuje slot připojení stále otevřený.

Pro úspěšný útok v mém případě stačilo toto jedno připojení. Odeslání již dvou souborů způsobilo pád serveru a jeho nedostupnost. Toto chování přisuzuji chybě v konkrétní implementaci serveru, nebo nesprávnému defaultnímu nastavení serveru. Pokud by však server toto chování nevykazoval, stačilo by, podobně jako u GET útoku, zvyšovat počet připojení až k dosáhnutí limitu serveru.

7.3 DNS útok

Na závěr jsem provedl test DNS útoku. Testovací útok jsem spouštěl s parametry, které jsou k nalezení v souboru `DNS-test.xml`. V tomto testu se otevřou tři vlákna, která se budou mého DNS serveru (`192.168.1.55`) dotazovat na přeložení adresy „root-servers.net“. V souboru `DNS-test.pcapng` nebo příloze C.3, můžeme vidět záznam testování tohoto útoku. Na záznamu můžeme vidět několik dvojic dotazů a odpovědí. Je zřejmé, že všechny části packetu jsou vytvořeny korektně. V prvním packetu, který je také zobrazen v příloze C.3, můžeme vidět odchycený DNS dotaz, v němž je požadována rekurze. Obsahuje jednu otázku na přeložení adresy „root-servers.net“. Důležité je všimnout si zdrojové IP adresy. Ta je podvrhnuta a nastavena na IP adresu oběti. IP adresa zdroje v packetu je `192.168.1.25`, přitom packet byl vytvářen a odeslán ze stroje s IP adresou `192.168.1.55`.

Úspěšný útok nepřikládám, protože z legálních důvodů nemohu použít veřejné DNS servery. Vhledem k tomu, že útoky spouštím ze stejného stroje, na kterém mi běží DNS server, není možno hovořit o odraženém útoku, a navíc by dříve došlo k zahlcení útočícího stroje, než k zahlcení oběti.

Kapitola 8

Závěr

Jedním z cílů práce bylo seznámit čtenáře s bezpečnostními riziky DoS útoků, jejich rozdělením a základními principy jejich činnosti. O této problematice pojednává 2. kapitola. Po porozumění činnosti těchto útoků a po prostudování dostupné literatury byly ve 3. kapitole navrženy možnosti ochrany proti těmto útokům. Ochrany preventivní, která by útokům měla zabránit, a ochrany aktivní, jež se dostává na řadu při právě probíhajícím útoku. Ve 4. kapitole jsme si ukázali, jak je v dnešní době jednoduché obstarat si funkční nástroj provádějící DoS útok. U nejznámějších DoS nástrojů je stručně popsána jejich historie a vysvětlen princip jejich činnosti. Dále jsou v této kapitole otestovány a zmíněny výhody, popř. nevýhody některých volně dostupných DoS nástrojů.

Pro prostudování dostupné literatury a otestování volně dostupných DoS nástrojů jsme s vedoucím práce dospěli k specifikaci zadání, která je popsána v kapitole 5. V této kapitole je také popsán návrh mnou implementovaného DoS nástroje, při kterém jsem se držel zadání, ale také jsem se snažil poučit z nedokonalostí, které jsem odhalil v testovaných nástrojích. V 6. kapitole je popsána implementace jádra nástroje i jednotlivých útoků a to především hierarchie vzájemného volání metod a řešení chybových stavů, které při útočení mohou nastat.

Můj DoS nástroj jsem po jeho implementaci testoval stejným způsobem, jako nástroje stažené. Můj nástroj je předčil především v tom, že jím generovaný útok úspěšně překonal firewall, kdežto např. u nástroje GoldenEye [22] ho pro úspěšný útok bylo třeba vypnout. Firewall byl překonán díky možnosti volit časové rozložení „odloženého startu“. Další výhodou mého nástroje je možnost přesně volit časové rozložení odesílaných packetů. Tato skutečnost sice testy v mých podmínkách neovlivnila, nicméně při útoku na sofistikovaněji chráněný systém by se tato výhoda jistě projevila. Testování mého nástroje je popsáno v kapitole číslo 7.

DoS nástroj jsem se samozřejmě snažil navrhnout a implementovat co nejlépe, nicméně není dokonalý. Pro jeho vylepšení by bylo možné lépe generovat náhodná data, odesílaná serveru při GET a POST útoku a to např. parametr pole `User-Agent:` v HTTP GET a POST hlavičce, jako je tomu např. u nástroje GoldenEye [22]. Dále by mohlo být výhodné u POST útoku umožnit odesílání skutečného souboru, namísto náhodně generovaných dat. Vítaným vylepšením by jistě bylo implementování kompatibility se systémem Windows, které jsem začal, ale bohužel nedokončil. Nejužitečnějším vylepšením pro bezpečnost, ale i funkčnost nástroje, bych označil možnost implementování některé anonymizační techniky, např. při GET a POST útoku umožnit připojování k oběti pomocí Proxy serverů nebo VPN.

Literatura

[1] Denial of Service Attacks. 1997 [cit. 2015-12-30].
URL http://www.cert.org/information-for/denial_of_service.cfm

[2] DNS Denial Of Service Tool. 2012 [cit. 2016-04-07].
URL https://packetstormsecurity.com/files/114890/DNS-Denial-Of-Service-Tool.html

[3] Alert (TA14-017A). 2014-2015 [cit. 2016-01-15].
URL https://www.us-cert.gov/ncas/alerts/TA14-017A

[4] DDoS Quick Guide. 2014 [cit. 2016-01-09].
URL https://www.us-cert.gov/sites/default/files/publications/DDoS%20Quick%20Guide.pdf

[5] DTD Tutorial. ©1999-2016 [cit. 2016-04-17].
URL http://www.w3schools.com/xml/xml_dtd_intro.asp

[6] TCP/IP Protocol Architecture. ©2016 [cit. 2016-01-09].
URL https://technet.microsoft.com/en-us/library/cc958821.aspx

[7] FENIX [online]. ©2016 [cit. 2016-04-02].
URL http://fe.nix.cz/

[8] R.U.D.Y. (R-U-Dead-Yet?). ©2016 [cit. 2016-04-06].
URL https://www.incapsula.com/ddos/attack-glossary/rudy-r-u-dead-yet.html

[9] Slowloris. ©2016 [cit. 2016-04-06].
URL https://www.incapsula.com/ddos/attack-glossary/slowloris.html

[10] Fielding, R. T.; Gettys, J.; Mogul, J. C.; aj.: Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, RFC Editor, June 1999 [cit. 2016-04-29], http://www.rfc-editor.org/rfc/rfc2616.txt.
URL http://www.rfc-editor.org/rfc/rfc2616.txt

[11] Kabelová, A.; Dostálek, L.: *Velký průvodce protokoly TCP/IP a systémem DNS*. Praha: Computer Press, třetí vydání, 2002 [cit. 2016-01-09], ISBN 80-722-6675-6.

[12] Kanapickas, P.; Stanley, M.: Std::function. 2011-2016 [cit. 2016-04-17].
URL http://en.cppreference.com/w/cpp/utility/functional/function

[13] Kanapickas, P.; Stanley, M.: Std::shared_ptr. 2011-2016 [cit. 2016-04-20].
URL http://en.cppreference.com/w/cpp/memory/shared_ptr

[14] Kočí, M.: 75letá důchodkyně překopala optický kabel, odpojila tím Gruzii a Arménii od internetu. In *PCTuning.cz [online]*, Praha: EMPRESA MEDIA, a.s., 2011 [cit. 2016-01-01], ISSN 1214-0201.
URL
http://pctuning.tyden.cz/component/content/article/1-aktualni-zpravy/20685-75leta-duchodkyne-prekopala-opticky-kabel-odpojila-tim-gruzii-a\-armenii-od-internetu

[15] Mockapetris, P.: Domain names - implementation and specification. STD 13, RFC Editor, November 1987 [cit. 2016-04-29], http://www.rfc-editor.org/rfc/rfc1035.txt.
URL http://www.rfc-editor.org/rfc/rfc1035.txt

[16] Moon, S.: Programming raw udp sockets in C on Linux. 2012 [cit. 2016-05-06].
URL http://www.binarytides.com/raw-udp-sockets-c-linux/

[17] Postel, J.: User Datagram Protocol. STD 6, RFC Editor, August 1980 [cit. 2016-04-29], http://www.rfc-editor.org/rfc/rfc768.txt.
URL http://www.rfc-editor.org/rfc/rfc768.txt

[18] Postel, J.: Internet Protocol. STD 5, RFC Editor, September 1981 [cit. 2016-04-29], http://www.rfc-editor.org/rfc/rfc791.txt.
URL http://www.rfc-editor.org/rfc/rfc791.txt

[19] Radware, L.: *DDoS Handbook*. Radware, Ltd., 2015 [cit. 2016-03-10], [online].
URL https://security.radware.com/

[20] Rossow, C.: Amplification Hell. In *NDSS Symposium [online]*, San Diego, USA: Internet Society, 2014 [cit. 2016-01-15], ISBN 1-891562-35-5.

[21] sammydre; Brennan, T.: ddos-toolbox. 2010-2014 [cit. 2016-04-07].
URL https://github.com/proactiveRISK/ddos-toolbox

[22] Seidl, J.: GoldenEye. 2012-2014 [cit. 2016-04-07].
URL https://github.com/jseidl/GoldenEye

[23] Stein, L.; Stewart, J.: The World Wide Web Security FAQ. 1995, 2003-04-23 [cit. 2015-12-30].
URL http://www.w3.org/Security/Faq/wwwsf6.html

[24] Tickle, A.; Ahmed, E.; Bhaskar, S.; aj.: Background. In *An Investigation into the Detection and Mitigation of Denial of Service (DoS) Attacks: Critical Information Infrastructure Protection*, editace S. Raghavan; E. Dawson, India: Springer, 2011 [cit. 2016-01-30], s. 9–40, doi:10.1007/978-81-322-0277-6_2.
URL http://eprints.qut.edu.au/80723/

[25] Veillard, D.: The XML C parser and toolkit of Gnome. [cit. 2016-04-20].
URL http://www.xmlsoft.org/index.html

[26] Willoner, E.; Roelofs, A.: DNS-Amplification-Attack. 2013-2015 [cit. 2016-04-07].
URL https://github.com/ethanwilloner/DNS-Amplification-Attack

Přílohy

Seznam příloh

Příloha A

Obsah CD

CD obsahuje následujících pět adresářů a jeden významově důležitý podadresář:

- `libs/` - obsahuje knihovny potřebné pro překlad a spuštění pod OS Windows
- `params/` - obsahuje ukázkové parametry pro spuštění útoků
- `pcap/` - obsahuje zaznamenané útoky generované mým DoS nástrojem
- `pcap/tools/` - obsahuje zaznamenané útoky generované staženými DoS nástroji
- `source/` - obsahuje zdrojový kód mého DoS nástroje
- `text/` - obsahuje zdrojový kód v LaTeXu pro generování této práce

CD dále obsahuje tyto dva dokumenty:

- `juhanak_BP.pdf` - elektronická verze této práce
- `manual.pdf` - dokumentace nástroje vygenerovaná nástrojem Doxygen

Adresář `source` také obsahuje DoS nástroj, pojmenovaný `dos_tool`, který je přeložený a spustitelný na školním serveru Merlin.

Příloha B

Manuál (README)

Copyright (c) 2016, Pavel Juhaňák

B.1 Licence

Tento zdrojový kód, jakožto i celý tento nástroj je dostupný pod OPEN SOURCE licencí VUT V BRNĚ.

Úplné znění licence dostupné v souboru `LICENSE.md`.

B.2 Info

Tento nástroj je programovou částí bakalářské práce na téma: Generování a ochrana proti DOS útoku na aplikační vrstvě. Tento nástroj umožňuje provádění tří DoS útoků a to HTTP GET útok, HTTP POST útok a DNS amplification útok.

Pro více informací si prosím přečtěte textovou část této práce obsaženou v souboru `juhanak_BP.pdf`, nebo dokumentaci vygenerovanou nástrojem Doxygen obsaženou v souboru `manual.pdf`.

B.3 Překlad

Pro standartní běh nástroje překládáme pomocí příkazu `make`. Pokročilejšího chování dosáhneme pomocí příkazu `make` společně s jedním z parametrů:

- `debug` - pro detailnější debugovací výpisy
- `rebuild` - pro smazání a znovupřeložení projektu
- `run` - pro běh s ukázkovými parametry
- `clean` - pro smazání souborů vzniklých při překladu

B.4 Spuštění

Nástroj je v současné době plně funkční pouze pod operačním systémem Linux.

Pro standartní běh je nástroji předáván jeden parametr a to XML soubor obsahující požadované parametry spuštění, validní podle DTD souboru `dos.dtd`. Ukázku validního

XML souboru s parametry můžeme nalézt v souboru `params.xml`, nebo v kterémkoliv ze souborů v adresáři `params`.

Pro výpis nápovědy nástroj spustíme s přepínačem `-h`, nebo `-help`.

Příklady spuštění:

- `dos_tool params.xml`
- `dos_tool -help`

B.5 Ukončení

Legitimní ukončení probíhá zasláním některého ze signálů:

- `SIGHUP`
- `SIGTERM`
- `SIGTSTP`

Signál `SIGINT` není odchytáván, aby bylo možné nástroj „násilně“ ukončit.

B.6 Návratové hodnoty

Hodnota	Situace
0	úspěch
1	chyba
2	tisk nápovědy

B.7 Závislosti

Pro správnou funkčnost nástroje je zapotřebí nainstalovat knihovnu `libxml2`, která je distribuována pod MIT licencí

- knihovna dostupná z: http://www.xmlsoft.org/index.html
- úplné znění licence https://opensource.org/licenses/mit-license.html

Linux

Knihovnu můžete nainstalovat podle návodu na stránkách knihovny, nebo zadáním následujících dvou příkazů, pokud používáte OS Ubuntu

```
sudo apt-get install libxml2-dev
sudo apt-get install libboost-all-dev
```

Windows

Binární soubory knihovny a jejích závislostí můžete získat ze stránek knihovny, nebo můžete použít binární soubory obsažené v adresáři `libs`, které jsou stažené právě z těchto stránek.

B.8 Převzatý kód

Generátor pseudonáhodných čísel

Generátor pseudonáhodných čísel deklarovaný v souboru `RandomGenerator.hpp`, jehož definice je obsažena v souboru `RandomGenerator.cpp` byl převzat ze společného projektu do předmětu *Modelování a simulace (IMS)*, vytvořeného v roce 2015. Autory tohoto projektu jsou **Jan Herec** a **Pavel Juhaňák** (já).

Funkce pro kontrolní součet

Funkce pro počítání kontrolního součtu `checksum()`, která je deklarována v souboru `DoS-lib.hpp` a jejíž definice je obsažena v souboru `main.cpp` byla převzata z webové stránky *BinaryTides*, konkrétně tutoriálu *Programming raw udp sockets in C on Linux*, dostupného z: http://www.binarytides.com/raw-udp-sockets-c-linux/. Tento tutoriál byl zveřejněn v roce 2012 a jeho autorem je uživatel pod pseudonymem **Silver Moon**.

B.9 ToDo

Implementace kompatibility s operačním systémem Windows.

Příloha C

Záznamy testů útoků

C.1 Záznam testu GET útoku

GET-test.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.55	192.168.1.20	TCP	74	52124 → 8080 [SYN] Seq=0 Win=29200 Len=0 MSS=146…
2	0.002482	192.168.1.20	192.168.1.55	TCP	74	8080 → 52124 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len…
3	0.003898	192.168.1.55	192.168.1.20	TCP	66	52124 → 8080 [ACK] Seq=1 Ack=1 Win=29312 Len=0 T…
4	0.505072	192.168.1.55	192.168.1.20	TCP	244	[TCP segment of a reassembled PDU]
5	0.505259	192.168.1.20	192.168.1.55	TCP	66	8080 → 52124 [ACK] Seq=1 Ack=179 Win=66560 Len=0…
6	1.007528	192.168.1.55	192.168.1.20	TCP	99	[TCP segment of a reassembled PDU]
7	1.007860	192.168.1.20	192.168.1.55	TCP	66	8080 → 52124 [ACK] Seq=1 Ack=212 Win=66560 Len=0…
8	1.511468	192.168.1.55	192.168.1.20	TCP	295	[TCP segment of a reassembled PDU]
9	1.511636	192.168.1.20	192.168.1.55	TCP	66	8080 → 52124 [ACK] Seq=1 Ack=441 Win=66304 Len=0…
10	1.512024	192.168.1.20	192.168.1.55	HTTP	580	HTTP/1.1 400 Bad Request (text/html)
11	1.512268	192.168.1.20	192.168.1.55	TCP	66	8080 → 52124 [FIN, ACK] Seq=515 Ack=441 Win=6630…
12	1.513997	192.168.1.55	192.168.1.20	TCP	66	52124 → 8080 [ACK] Seq=441 Ack=515 Win=30336 Len…
13	1.579393	192.168.1.55	192.168.1.20	TCP	66	52124 → 8080 [ACK] Seq=441 Ack=516 Win=30336 Len…
14	2.015736	192.168.1.55	192.168.1.20	TCP	244	[TCP segment of a reassembled PDU]
15	2.015872	192.168.1.20	192.168.1.55	TCP	66	8080 → 52124 [ACK] Seq=516 Ack=619 Win=66048 Len…
16	2.515921	192.168.1.55	192.168.1.20	TCP	235	[TCP segment of a reassembled PDU]
17	2.516281	192.168.1.20	192.168.1.55	TCP	66	8080 → 52124 [ACK] Seq=516 Ack=788 Win=65792 Len…
18	2.915074	192.168.1.55	192.168.1.20	TCP	97	[TCP segment of a reassembled PDU]
19	2.915219	192.168.1.20	192.168.1.55	TCP	66	8080 → 52124 [ACK] Seq=516 Ack=819 Win=65792 Len…
20	2.915504	192.168.1.55	192.168.1.20	TCP	66	52124 → 8080 [FIN, ACK] Seq=819 Ack=516 Win=3033…
21	2.915596	192.168.1.20	192.168.1.55	TCP	66	8080 → 52124 [ACK] Seq=516 Ack=820 Win=65792 Len…

```
▷ Frame 4: 244 bytes on wire (1952 bits), 244 bytes captured (1952 bits) on interface 0
▷ Ethernet II, Src: CadmusCo_b5:98:c2 (08:00:27:b5:98:c2), Dst: LiteonTe_e7:fe:92 (28:e3:47:e7:fe:92)
▷ Internet Protocol Version 4, Src: 192.168.1.55, Dst: 192.168.1.20
▷ Transmission Control Protocol, Src Port: 52124 (52124), Dst Port: 8080 (8080), Seq: 1, Ack: 1, Len: 178

0000  28 e3 47 e7 fe 92 08 00  27 b5 98 c2 08 00 45 00   (.G..... '.....E.
0010  00 e6 3b 9a 40 00 40 06  7a dc c0 a8 01 37 c0 a8   ..;.@.@. z....7..
0020  01 14 cb 9c 1f 90 b1 93  af a5 59 71 98 45 80 18   ........ ..Yq.E..
0030  00 e5 bc d7 00 00 01 01  08 0a 00 22 94 ba 01 d3   ........ ..."....
0040  73 e7 47 45 54 20 2f 20  48 54 54 50 2f 31 2e 31   s.GET /  HTTP/1.1
0050  0d 0a 48 6f 73 74 3a 20  31 39 32 2e 31 36 38 2e   ..Host:  192.168.
0060  31 2e 32 30 0d 0a 55 73  65 72 2d 41 67 65 6e 74   1.20..Us er-Agent
0070  3a 20 4d 6f 7a 69 6c 6c  61 2f 35 2e 30 20 28 63   : Mozill a/5.0 (c
0080  6f 6d 70 61 74 69 62 6c  65 3b 20 4d 53 49 45 20   ompatibl e; MSIE
0090  31 30 2e 30 3b 20 57 69  6e 64 6f 77 73 20 4e 54   10.0; Wi ndows NT
00a0  20 37 2e 30 3b 20 49 6e  66 6f 50 61 74 68 2e 33    7.0; In foPath.3
00b0  3b 20 2e 4e 45 54 20 43  4c 52 20 33 2e 31 2e 34   ; .NET C LR 3.1.4
00c0  30 37 36 37 3b 20 54 72  69 64 65 6e 74 2f 36 2e   0767; Tr ident/6.
00d0  30 3b 20 65 6e 2d 49 4e  29 0d 0a 43 61 63 68 65   0; en-IN )..Cache
```

C.2 Záznam testu POST útoku

POST-test.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.55	192.168.1.20	TCP	74	57990 → 8080 [SYN] Seq=0 Win=29200 Len=0 MSS=14…
2	0.000479	192.168.1.20	192.168.1.55	TCP	74	8080 → 57990 [SYN, ACK] Seq=0 Ack=1 Win=8192 Le…
3	0.003378	192.168.1.55	192.168.1.20	TCP	66	57990 → 8080 [ACK] Seq=1 Ack=1 Win=29312 Len=0 …
4	0.503625	192.168.1.55	192.168.1.20	TCP	346	[TCP segment of a reassembled PDU]
5	0.504098	192.168.1.20	192.168.1.55	TCP	66	8080 → 57990 [ACK] Seq=1 Ack=281 Win=66560 Len=…
6	1.011085	192.168.1.55	192.168.1.20	TCP	130	[TCP segment of a reassembled PDU]
7	1.011225	192.168.1.20	192.168.1.55	TCP	66	8080 → 57990 [ACK] Seq=1 Ack=345 Win=66304 Len=…
8	1.520011	192.168.1.55	192.168.1.20	TCP	130	[TCP segment of a reassembled PDU]
9	1.520336	192.168.1.20	192.168.1.55	TCP	66	8080 → 57990 [ACK] Seq=1 Ack=409 Win=66304 Len=…
10	2.028818	192.168.1.55	192.168.1.20	TCP	130	[TCP segment of a reassembled PDU]
11	2.029142	192.168.1.20	192.168.1.55	TCP	66	8080 → 57990 [ACK] Seq=1 Ack=473 Win=66304 Len=…
12	2.543024	192.168.1.55	192.168.1.20	TCP	129	[TCP segment of a reassembled PDU]
13	2.543369	192.168.1.20	192.168.1.55	TCP	66	8080 → 57990 [ACK] Seq=1 Ack=536 Win=66304 Len=…
14	2.939869	192.168.1.20	192.168.1.55	TCP	1514	[TCP segment of a reassembled PDU]
15	2.939886	192.168.1.20	192.168.1.55	TCP	1514	8080 → 57990 [ACK] Seq=1449 Ack=536 Win=66304 L…
16	2.939894	192.168.1.20	192.168.1.55	TCP	1514	8080 → 57990 [ACK] Seq=2897 Ack=536 Win=66304 L…

```
▷ Frame 4: 346 bytes on wire (2768 bits), 346 bytes captured (2768 bits) on interface 0
▷ Ethernet II, Src: CadmusCo_b5:98:c2 (08:00:27:b5:98:c2), Dst: LiteonTe_e7:fe:92 (28:e3:47:e7:fe:92)
▷ Internet Protocol Version 4, Src: 192.168.1.55, Dst: 192.168.1.20
▷ Transmission Control Protocol, Src Port: 57990 (57990), Dst Port: 8080 (8080), Seq: 1, Ack: 1, Len: 280

0000  28 e3 47 e7 fe 92 08 00  27 b5 98 c2 08 00 45 00   (.G..... '.....E.
0010  01 4c 58 99 40 00 40 06  5d 77 c0 a8 01 37 c0 a8   .LX.@.@. ]w...7..
0020  01 14 e2 86 1f 90 6c 0b  d4 69 a9 fc 0d b4 80 18   ......l. .i......
0030  00 e5 6f b6 00 00 01 01  08 0a 00 04 8d 7d 00 a3   ..o..... .....}..
0040  68 3f 50 4f 53 54 20 2f  20 48 54 54 50 2f 31 2e   h?POST /  HTTP/1.
0050  31 0d 0a 48 6f 73 74 3a  20 31 39 32 2e 31 36 38   1..Host:  192.168
0060  2e 31 2e 32 30 0d 0a 55  73 65 72 2d 41 67 65 6e   .1.20..U ser-Agen
0070  74 3a 20 4d 6f 7a 69 6c  6c 61 2f 35 2e 30 20 28   t: Mozil la/5.0 (
0080  63 6f 6d 70 61 74 69 62  6c 65 3b 20 4d 53 49 45   compatib le; MSIE
0090  20 31 30 2e 30 3b 20 57  69 6e 64 6f 77 73 20 4e    10.0; W indows N
00a0  54 20 37 2e 30 3b 20 49  6e 66 6f 50 61 74 68 2e   T 7.0; I nfoPath.
00b0  33 3b 20 2e 4e 45 54 20  43 4c 52 20 33 2e 31 2e   3; .NET  CLR 3.1.
00c0  34 30 37 36 37 3b 20 54  72 69 64 65 6e 74 2f 36   40767; T rident/6
00d0  2e 30 3b 20 65 6e 2d 49  4e 29 0d 0a 43 61 63 68   .0; en-I N)..Cach
00e0  65 2d 43 6f 6e 74 72 6f  6c 3a 20 6e 6f 2d 63 61   e-Contro l: no-ca
00f0  63 68 65 0d 0a 43 6f 6e  6e 65 63 74 69 6f 6e 3a   che..Con nection:
0100  20 6b 65 65 70 2d 61 6c  69 76 65 0d 0a 43 6f 6e    keep-al ive..Con
0110  74 65 6e 74 2d 54 79 70  65 3a 20 61 70 70 6c 69   tent-Typ e: appli
0120  63 61 74 69 6f 6e 2f 78  2d 77 77 77 2d 66 6f 72   cation/x -www-for
0130  6d 2d 75 72 6c 65 6e 63  6f 64 65 64 0d 0a 43 6f   m-urlenc oded..Co
0140  6e 74 65 6e 74 2d 4c 65  6e 67 74 68 3a 20 32 35   ntent-Le ngth: 25
0150  35 0d 0a 0d 0a 66 69 6c  65 3d                    5....fil e=
```

C.3 Záznam testu DNS útoku

DNS-test.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.25	192.168.1.55	DNS	78	Standard query 0xd1e4 ANY root-servers.net
2	0.000353	192.168.1.55	192.168.1.25	DNS	542	Standard query response 0xd1e4 ANY root-serve…
3	2.316999	192.168.1.25	192.168.1.55	DNS	78	Standard query 0x6858 ANY root-servers.net
4	2.317840	192.168.1.55	192.168.1.25	DNS	542	Standard query response 0x6858 ANY root-serve…
5	2.820670	192.168.1.25	192.168.1.55	DNS	78	Standard query 0x26d9 ANY root-servers.net
6	2.821092	192.168.1.55	192.168.1.25	DNS	542	Standard query response 0x26d9 ANY root-serve…
7	4.964931	192.168.1.25	192.168.1.55	DNS	78	Standard query 0x32ee ANY root-servers.net
8	4.965558	192.168.1.55	192.168.1.25	DNS	542	Standard query response 0x32ee ANY root-serve…
9	5.059652	192.168.1.25	192.168.1.55	DNS	78	Standard query 0x14b8 ANY root-servers.net

```
▷ Frame 1: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface 0
▷ Linux cooked capture
▷ Internet Protocol Version 4, Src: 192.168.1.25, Dst: 192.168.1.55
▷ User Datagram Protocol, Src Port: 8080 (8080), Dst Port: 53 (53)
◢ Domain Name System (query)
    [Response In: 2]
    Transaction ID: 0xd1e4
  ◢ Flags: 0x0100 Standard query
      0... .... .... .... = Response: Message is a query
      .000 0... .... .... = Opcode: Standard query (0)
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... .0.. .... = Z: reserved (0)
      .... .... ...0 .... = Non-authenticated data: Unacceptable
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ◢ Queries
    ◢ root-servers.net: type ANY, class IN
        Name: root-servers.net
        [Name Length: 16]
        [Label Count: 2]
        Type: * (A request for all records the server/cache has available) (255)
        Class: IN (0x0001)
```

```
0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ........ ........
0010  45 00 00 3e 75 24 00 00  ff 11 c2 e9 c0 a8 01 19   E..>u$.. ........
0020  c0 a8 01 37 1f 90 00 35  00 2a f7 87 d1 e4 01 00   ...7...5 .*......
0030  00 01 00 00 00 00 00 00  0c 72 6f 6f 74 2d 73 65   ........ .root-se
0040  72 76 65 72 73 03 6e 65  74 00 00 ff 00 01         rvers.ne t.....
```