UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Simulace útoků na síťovou infrastrukturu

Radek Knytl

Bakalářská práce
2014

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2013/2014**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Radek Knytl**
Osobní číslo:	**I11095**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Simulace útoků na síťovou infrastrukturu**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Autor se ve své práci bude zabývat síťovými útoky a jejich simulací pomocí nástroje Imalse. V teoretické části popíše nejdůležitější síťové útoky a nastíní možná opatření proti těmto útokům. Dále provede podrobné představení funkcionalit nástroje Imalse (Integrated Malware Simulator and Emulator). V praktické části práce nástroj Imalse nasadí a zrealizuje na namodelované síťové infrastruktuře moderní síťové útoky. Dále v práci představí výstupy poskytnuté nástrojem Imalse a provede jejich analýzu.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

ENGEBRETSON, Pat a James BROAD. The basics of hacking and penetration testing: ethical hacking and penetration testing made easy. Waltham, MA: Syngress, c2011, xvii, 159 p. ISBN 15-974-9655-3.
HERZOG, Pete. ISECOM. Open source security testing methodology manual. 3. vyd. 2010. Dostupné z: http://www.isecom.org/mirror/OSSTMM.3.pdf

Vedoucí bakalářské práce: **Ing. Soňa Neradová**
Katedra softwarových technologií

Datum zadání bakalářské práce: **20. prosince 2013**
Termín odevzdání bakalářské práce: **9. května 2014**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2014

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 2. 5. 2014

Radek Knytl

Poděkování

Rád bych tímto poděkoval své vedoucí práce, Ing. Soně Neradové, za ochotu, přínosné rady a věnovaný čas, při realizaci této práce a Mgr. Josefu Horálkovi, Ph.D. za cenné rady při výběru tématu. Také bych chtěl poděkovat svým rodičům, kteří mě podporovali během studia a psaní této bakalářské práce.

Anotace

Tato bakalářská práce se zabývá simulacemi a analýzou vybraných síťových útoků na jednotlivých vrstvách ISO/OSI modelu. Dále je popsán princip provádění, obrana a možná prevence proti těmto útokům.

Klíčová slova

síťové útoky, síťová bezpečnost, Imalse, DoS, simulace útoků, firewall

Title

Simulations of attacks on network infrastructure

Annotation

This bachelor’s thesis deals with simulations and analysis of selected network attacks at each layer of ISO/OSI model. It also describes the implementation of the principle, defense and possible prevention against these attacks.

Keywords

network attacks, network security, Imalse, DoS, simulations of attacks, firewall

Obsah

Seznam zkratek

ACL	Access Control List
AH	Authentication Header
ARP	Address Resolution Protocol
BGP	Border Gateway Protocol
CORE	Common Open Research Emulator
DDoS	Distributed Denial of Service
DoS	Denial of Service
DNS	Domain Name System
ESP	Encapsulating Security Payload
FTP	File Transfer Protocol
GUI	Graphical User Interface
HSTS	HTTP Strict Transport Security
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IKE	Internet Key Exchange
IMALSE	Integrated Malware Simulator and Emulator
IP	Internet Protocol
IPS	Instruction Prevention System
IPsec	Internet Protocol security
ISO/OSI	International Standards Organization/Open System Interconnection
MAC	Media Access Control
MAC	Message Authentication Code
MD5	Message Digest Algorithm
MITM	Man In The Middle
OSPF	Open Shortest Path First
RIP	Routing Information Protocol
SMTP	Simple Mail Transfer Protocol
SA	Security Associations
SHA	Secure Hash Algorithm
SPA	Security Policy Associations
SPD	Security Policy Database
SPI	Security Parameter Index
SSL	Secure Sockets Layer
TCP/IP	Transmission Control Protocol/Internet Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
VM	Virtual Machine
VoIP	Voice over Internet Protocol

Seznam obrázků

Seznam tabulek

Úvod

V dnešní době si již běžné fungování společnosti bez informačních technologií nelze představit. Prakticky všechna odvětví lidské činnosti jsou závislá, ať už ve větší či menší míře, na používání osobních počítačů, notebooků, tabletů, chytrých mobilních telefonů a dalších zařízení založených na informačních technologiích. S tím také úzce souvisí jejich vzájemná komunikace, přenos a uchovávání dat. V podstatě dnes existuje jen velmi málo zařízení, která by fungovala zcela izolovaně bez nutnosti připojení ke komunikační síti.

Komunikační sítě nám umožňují téměř neomezené možnosti, jak nakládat s informacemi. S tím také vyvstává hrozba, která vznikla společně s prvním zpracováváním a uchováváním informací, několik tisíc let před naším letopočtem, a to snaha o neoprávněné získání těchto informací za různými účely.

Z historie je známo spousta takovýchto případů, ať už se jednalo o uloupení tajných spisů či dešifrování zpráv pro vojenské účely. Myšlenka zmocnit se citlivých informací zůstává, pouze se informace v tomto případě přesněji data transformovala z papírů do kybernetického světa.

Počítačové útoky jsou z převážné většiny podnikány právě za účelem získání citlivých dat, dalšími důvody pak mohou být poškození oběti, například konkurenční společností, vyjádření nesouhlasu nebo upozornění na bezpečnostní slabiny. Z globálního hlediska se jedná o dosti diskutované a řešené téma, kterému je přikládána stále větší váha. Každý dobrý správce sítě se snaží mít svou síť co nejlépe chráněnou tak, aby byla data uvnitř bezpečně chráněna. Nikdy však nelze zaručit naprostou bezpečnost. Vždy existuje způsob, jak lze dané zabezpečení prolomit. Při zabezpečování sítí a systémů se tak volí úroveň zabezpečení zejména podle ceny, jakou chráněná data představují.

Teoretická část práce popisuje základní rozdělení útoků, charakteristiku jednotlivých komunikačních vrstev síťového modelu ISO/OSI a TCP/IP. Následuje popis variant možných útoků na těchto vrstvách. Podrobně jsou také rozepsány principy nejdůležitější bezpečnostních protokolů používaných pro zabezpečení komunikace. Poslední kapitola teoretické části se věnuje popisu obecných zásad a prostředků pro zabezpečení síťové infrastruktury.

V praktické části jsou nejprve představeny nástroje a prostředí pro simulaci a popis jejich instalací, zejména nástroj Imalse, jenž se využívá pro simulování sítě botnet. V následných krocích je popsáno provedení simulace sítě botnet a příslušných útoků za použití Imalse. Dále je představena simulace vybraných útoků pomocí nástrojů v operačním systému Kali Linux, sloužících k penetračnímu testování na fyzických zařízeních. Důležitou součást praktické části zahrnuje popis principů, možností obrany a prevence proti simulovaným útokům. V závěru práce jsou zhodnoceny poznatky získané z provedených simulací pomocí výše uvedených nástrojů.

1 Síťové útoky

1.1 Definice síťového útoku

Pojem síťový útok lze chápat jako cílené nelegální provádění aktivit za pomocí počítačových sítí vedoucí k proniknutí do privátních sítí nebo počítačových systémů, připojených k síti, za různými účely. Například odcizení/poškození soukromých informací, znepřístupnění (odstavení) síťových služeb, zařazení do botnetu[1]. Při provádění útoků využívá útočník znalostí architektury kompromitované sítě a jejich slabých míst jako jsou nedokonalosti protokolů či chyby aplikací (např. neošetření vstupních dat).

1.2 Rozdělení síťových útoků

Na rozdělení síťových útoků lze nahlížet z několika aspektů:

1.2.1 Vnitřní, vnější útoky

- **Vnitřní útoky** – Útočník se při těchto útocích nachází uvnitř kompromitované sítě, může jít o zaměstnance, který se snaží zaútočit na firemní intranet. Využít lze například bezpečnostních slabin aplikací, login spoofing, slovníkový útok pro získání administrátorských práv.

- **Vnější útoky** – Útočník přistupuje k veřejně běžícím službám z vnějšího prostředí na základě adresy IP pro daný server. Při útocích se využívá bezpečnostních chyb či slabých míst v daných službách.

Více o vnitřních a vnějších útocích se lze dočíst v (Dočekal, 2010).

1.2.2 Útoky podle vrstev modelu ISO/OSI

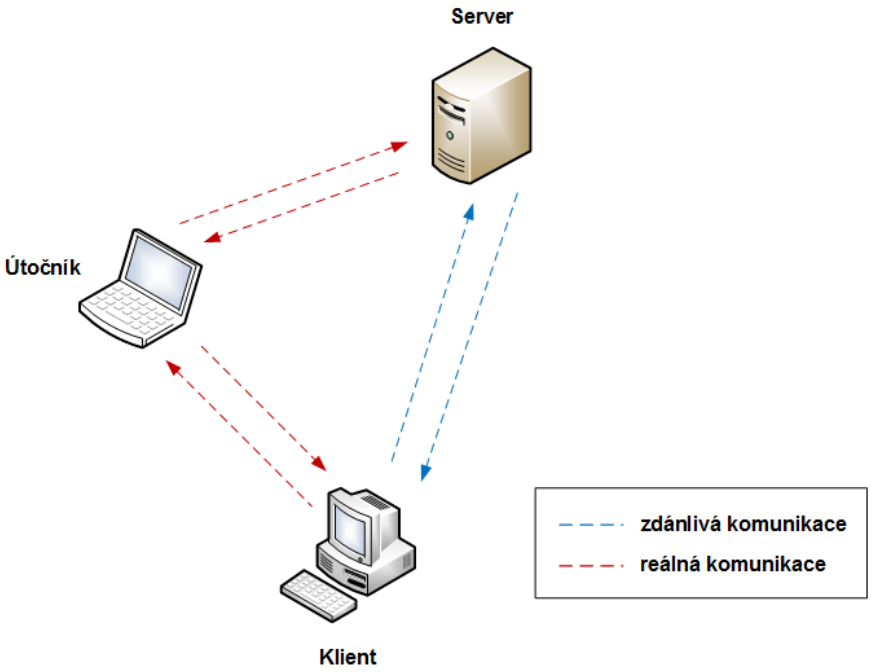
Jednotlivé vrstvy modelu ISO/OSI v sobě zahrnují protokoly. V době kdy většina těchto protokolů vznikala, nebyl kladen příliš velký důraz na bezpečnostní politiku. Díky tomu útočníci využívají nedokonalostí a slabin těchto protokolů k podnikání různých typů útoků na jednotlivých vrstvách, jejichž příklady jsou uvedeny v kapitole 2.2.

1.2.3 Útoky typu Man in the middle (MITM)

Základní myšlenka útoků typu Man in the middle spočívá v tom, že se útočník stane prostředníkem v komunikaci mezi dvěma zařízeními. V takovém případě může odposlouchávat takovou komunikaci a získávat tak v mnohých případech citlivá data nebo dokonce komunikaci modifikovat a cílovému zařízení odesílat podvržená data, vizte Obrázek 1.1.

Mezi nejznámější útoky tohoto typu patří ARP poisoning, DNS spoofing, SSL spoofing, ICMP redirection, DHCP snooping.

[1] Síť infikovaných počítačů (softwarovými agenty/boty). Botnet pak umožnuje útočníkovi využít těchto infikovaných počítačů (bez vědomí jejich uživatelů) k nelegálním činnostem jako rozesílání spamu či DDoS útokům.



Obrázek 1.1 – Schéma útoku Man in the middle

1.2.4 Útoky typu Denial of Service (DoS)

Při útocích typu Denial of Service, které patří k jedněm z nejúčinnějších, je cílem útočníka dostat server či jiné síťové zařízení do stavu, kdy dojde k odmítnutí přístupu k jejich službám nebo prostředkům pro ostatní oprávněné klienty (Obrázek 1.2).

Útoky se tedy útočník nesnaží odcizit data, ale restartovat nebo odstavit server a v závislosti na jeho konfiguraci může tato skutečnost vést například k poškození systému případně jeho aplikací, jak píše Nelson (2011).

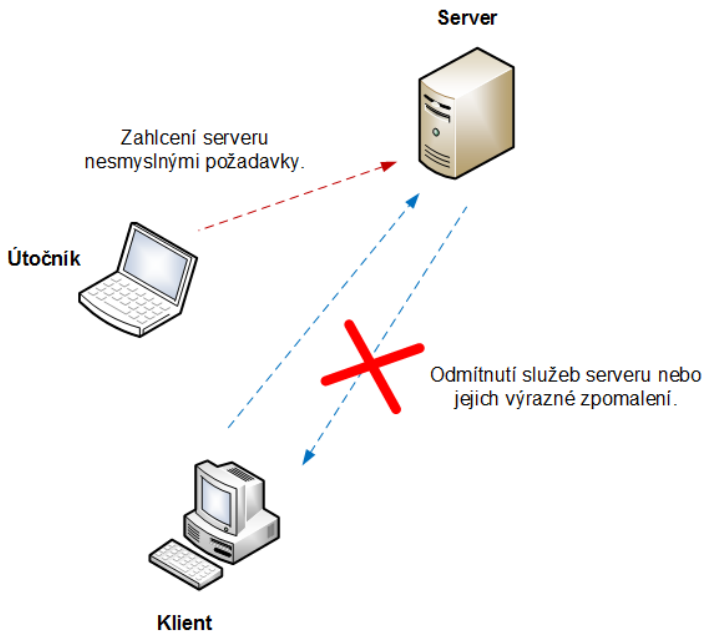
Realizace útoků je prováděna pomocí následujících čtyř scénářů (jako oběť útoku bude uvažován server), popsaných v (Přibyl, 2006):

- **Obsazením přenosové kapacity** – Útočník se snaží vytvořit takový provoz směrem k serveru, při kterém v co největší míře zatížit případně zcela zahltí přenosovou linku, čímž zabrání dalším uživatelům využívat této linky. K zatížení linky serveru lze využít ze strany útočníka linku o vyšší kapacitě nebo spojení několika linek o menší kapacitě.
- **Spotřebování limitovaných zdrojů** – Podstata metody spočívá v tom, že útočník spotřebovává zdroje serveru například paměť nebo čas procesoru. Dojde tedy k úplnému vyčerpání nebo co nejvyššímu využití zmíněných zdrojů, což má za následek opět snížení kvality služeb serveru, případně jejich úplné odmítnutí pro ostatní klienty.
- **Zneužití chyb v aplikaci** – U této metody využívá útočník chyb v aplikacích serveru (nejčastěji opomenutím aktualizace dané aplikace), kdy server na neobvyklou situaci reaguje nestandardním chováním aplikace (např. zacyklením, pádem).

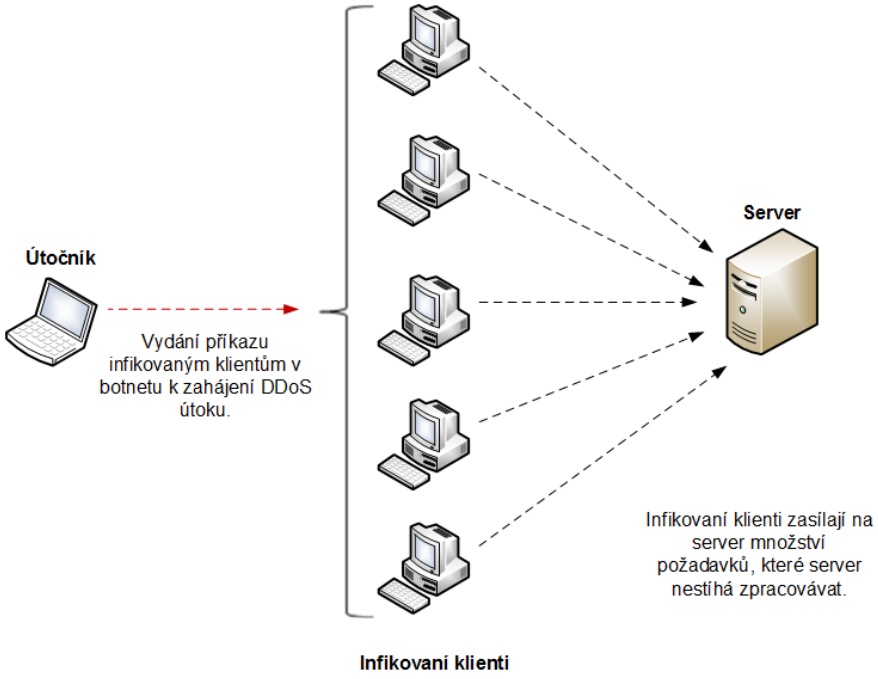
- **Napadení DNS a systémů směrování paketů** – Metoda umožnuje útočníkovi změnit záznamy o adresách IP v serveru DNS. Díky tomu je pak veškerý provoz původně směřovaný k legitimnímu cíli (např. webový server) modifikován na útočníkem podvržený cíl.

Podkategorií útoku DoS je takzvaný distribuovaný útok DoS neboli DDoS. Zásadním rozdílem je především počet klientů zahlcujících server požadavky. Na útoku DDoS se nevědomě podílí velké množství (stovky až tisíce) infikovaných klientů zařazených do botnetu centrálně řízených útočníkem, tuto situaci zachycuje Obrázek 1.3.

Konkrétními příklady útoků jsou ICMP (Ping) flood, SYN flood, Ping of Death, DNS Amplification.



Obrázek 1.2 – Schéma útoku Denial of Service



Obrázek 1.3 – Schéma útoku Distributed Denial of Service

1.2.5 Útoky typu spoofing

Útoky typu spoofing jsou založené na technice, kdy útočná strana maskuje svou pravou identitu a vydává se za jiného uživatele či zařízení v síti za pomoci falšování dat. Díky čemuž poté získá neoprávněný přístup do sítí či systémů obětí. Útočník tak zde může nerušeně odcizovat data, šířit malware[2] a provádět další nelegální činnosti.

- **Phishing útoky** – Často se také označují jako „rhybaření“. Jedná se o formu útoků spadající také do kategorie sociálního inženýrství, vizte (McDowell, 2009). Jak již název napovídá, je zde analogie s rybařením, kdy stejně tak se počítačový útočník snaží pomocí návnady „ulovit oběť“ (získat tajné informace). Využívá k tomu nezkušeností obětí s danou problematikou a schopnosti přesvědčení k vykonání patřičných úkonů (např. přístup k falešným webovým stránkám a zadání přístupových údajů). Phishing útoky jsou podmnožinou útoků typu spoofing.

Ke známým příkladům útoků tohoto typu patří například IP spoofing, ARP spoofing, DNS spoofing.

1.2.6 Viry, červi, trojské koně, boti

- **Viry** – Označení pro škodlivé kódy, založené na stejném principu jako biologické viry. K jejich aktivaci dochází prostřednictvím spuštění infikovaného programu uživatelem, což je jedna z hlavních odlišností od červů. Poté, co jsou viry aktivovány, šíří své kódy do jiných programů, provádí destrukce dat či jejich kompromitaci. Na

[2] Škodlivý program, určený k poškozování systémů, průnikům útočníka do systémů, zasílání kompromitovaných informací obětí útočníkovi. Vizte kapitola 1.2.6.

ostatní stanice jsou viry šířeny prostřednictvím infikovaných programů, například odesláním přes e-mail, stažením ze serveru FTP atd.

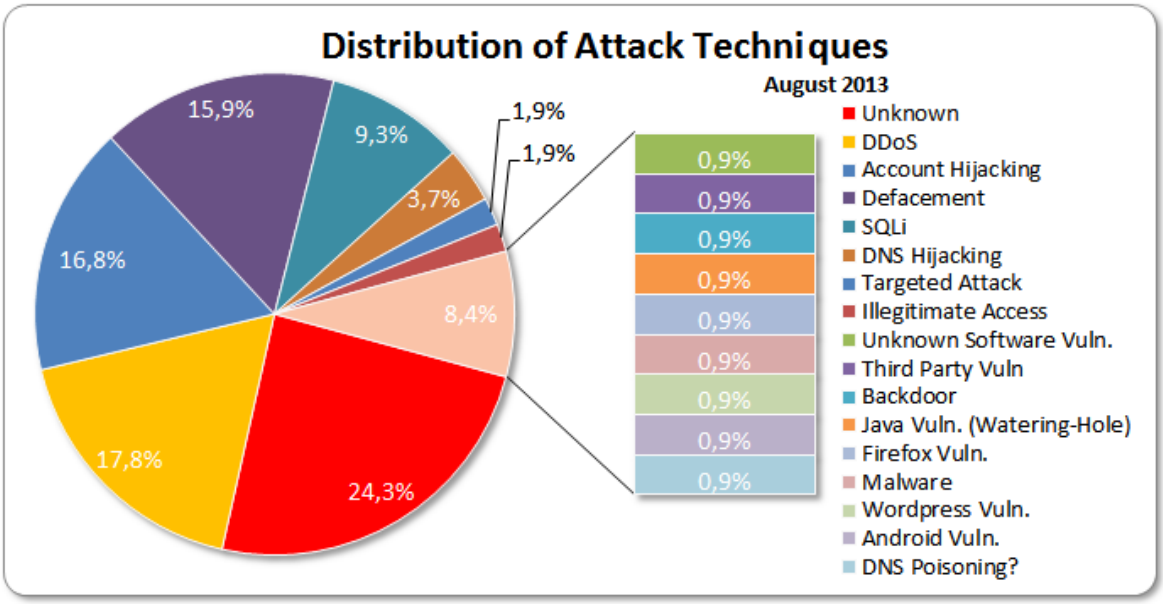
- **Červi** – Rovněž se jedná o škodlivé kódy, ovšem na rozdíl od virů neprobíhá jejich aktivace lidským přičiněním. Spouští se automaticky s operačním systémem (naplánováním jejich spuštění po startu OS). Dalším rozdílem je fakt, že se nevkládají do hostitelských programů, ale jejich kódy jsou uloženy přímo v operační paměti nebo na sekundární paměti, kde vytvářejí samostatné soubory. Dokáží se samostatně šířit i sítí.

- **Trojské koně** – Jak již z mytologie vyplývá, trojské koně se uživateli jeví jako přátelské (užitečné) programy, například počítačové hry, spořiče obrazovky, komunikační programy apod. Při spuštěni těchto programů dochází k vykonávání vedlejší škodlivé činnosti, aniž by o nich uživatel věděl. Škodlivé části programů uvnitř trojských koňů nejsou schopny se samostatně rozšiřovat do jiných programů ani počítačů.

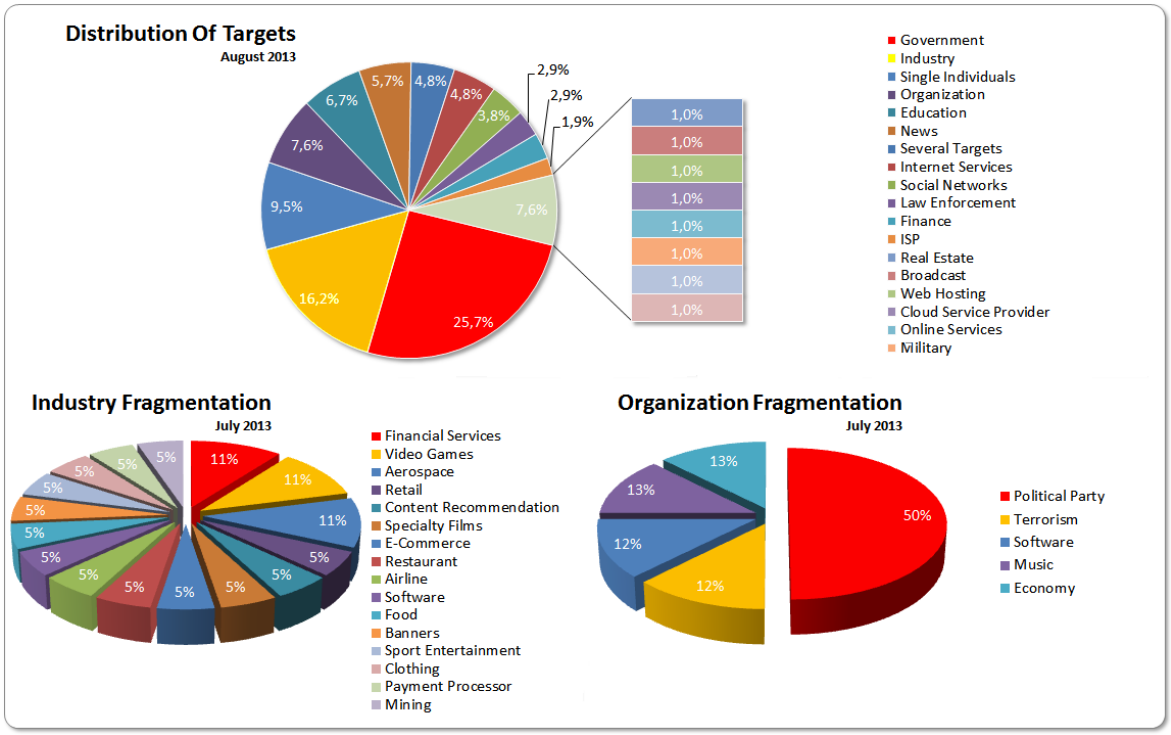
- **Boti** – Představují druh malware, který napadá počítače uživatelů a dává útočníkovi veškerou kontrolu nad takto infikovanou stanicí. Stejně jako červi se dokáží šířit samostatně. Ovládají různé procesy a služby operačního systému, aniž by o tom napadený uživatel věděl. Mohou tak například prohledávat soubory a hledat citlivé informace (např. hesla), analyzovat pakety, zaznamenávat stisknuté klávesy a mnohé další úkony. Nejčastěji se však bot využívá k napadení stanice a jejímu připojení k centrálnímu serveru. Tento server poté slouží k řízení všech infikovaných stanic, které jsou k němu připojeny. Vzniká tak síť infikovaných stanic, neboli botnet. S botnetem souvisí ještě tzv. botmaster, což je stanice útočníka, která zasílá serveru požadavky k zahájení, řízení průběhu a ukončení útoků typu DDoS, popsaných v 1.2.4.

Více k tomuto tématu se lze dočíst v (Cisco Systems Inc., 2009; Holub, 2002).

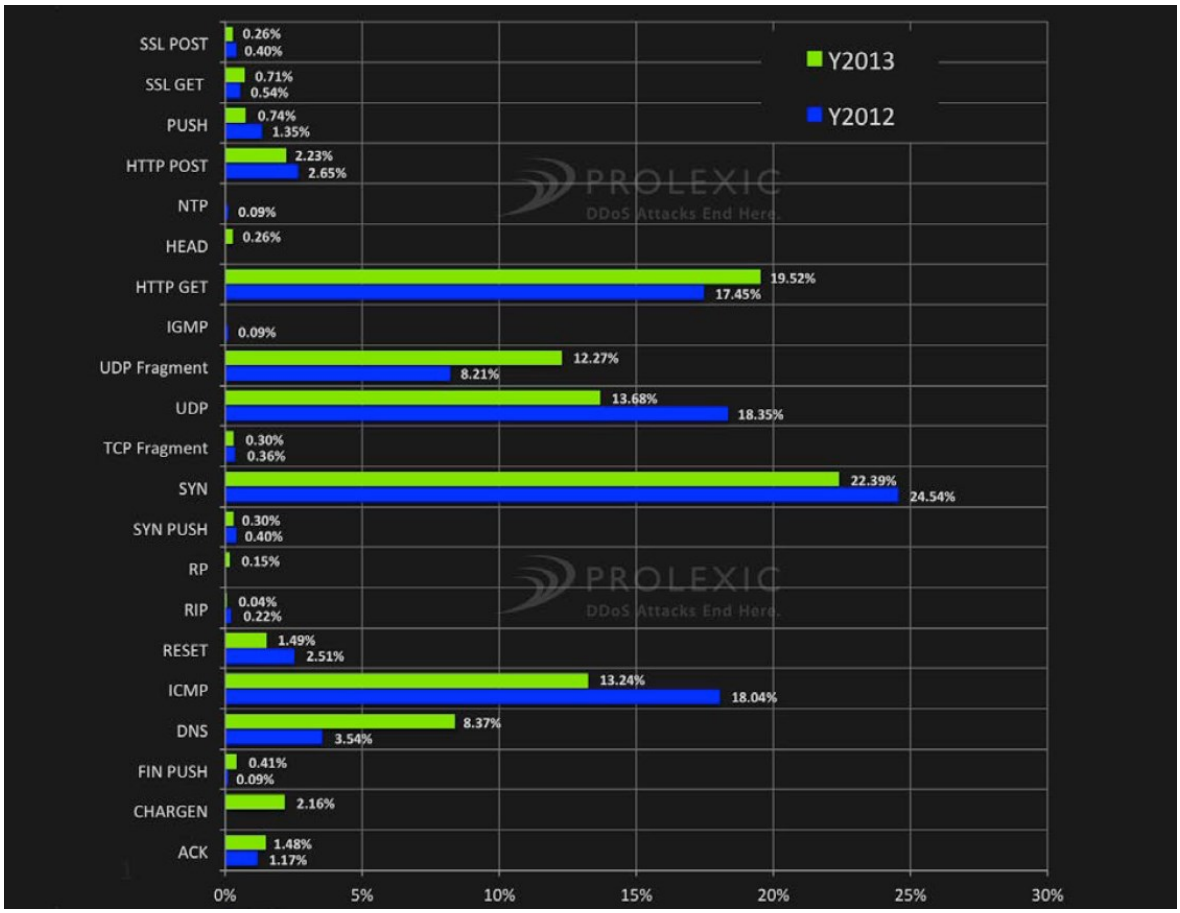
1.3 Statistiky síťových útoků



Obrázek 1.4 – Graf: Rozdělení technik útoků (Passeri, 2013)



Obrázek 1.5 – Grafy útoků: Rozdělení podle cílů; Dělení podle průmyslu; Dělení podle organizací (Passeri, 2013)



Obrázek 1.6 – Graf porovnání realizovaných útoků typu DDoS (Prolexic Technologies Inc., 2014)

2 Model ISO/OSI

Referenční model ISO/OSI byl vyvinut organizací ISO[3] v 80. letech minulého století za účelem poskytnutí otevřené síťové architektury a vytvoření jednotné podoby sítě založené na této architektuře. Do té doby si většina velkých firem vytvářela vlastní proprietární síťové architektury, které neumožňovaly komunikaci systémů se systémy jiných výrobců.

Jedná se tedy o konceptuální sedmi vrstvý model, obsahující soubor standardů pro definici chování jednotlivých vrstev a pravidel pro komunikaci síťových zařízení splňující tyto normy. Každá z těchto vrstev poskytuje služby pro svou horní sousední vrstvu (s výjimkou nejvyšší vrstvy). Horní tři vrstvy pak umožnují komunikaci aplikací s uživatelem a dolní čtyři vrstvy zajišťují přenos dat z jedné stanice na druhou.

Obrázek 2.1 – Model ISO/OSI

2.1 Popis vrstev

- **Aplikační vrstva** – Tato vrstva vytváří rozhraní mezi aplikací a nižší sousední vrstvou. Vrstva je aktivována ve chvíli, kdy aplikace požaduje komunikaci za použití protokolů, jež jí náleží. Také se stará o identifikaci a dostupnost zařízení, se kterým má komunikovat. Mezi nejčastěji nabízené služby patří přístup k webovým stránkám, zasílání zpráv, přenos souborů.

- **Prezentační vrstva** – Úkolem prezentační vrstvy je překlad dat při jejich odesílání z formátu konkrétní aplikace na obecný formát, který není závislý na konkrétní platformě. V případě přijímání dat probíhá tento proces opačným způsobem. Dalšími službami jsou komprese a šifrování/dešifrování dat.

[3] Mezinárodní normalizační organizace (International Standards Organization)

- **Relační vrstva** – Relační vrstva vytváří, řídí a ukončuje přenosy dat mezi uzly. Řízení komunikace mezi uzly je prováděno v jednom ze tří režimů simplex, half-duplex, full-duplex.

- **Transportní vrstva** – V transportní vrstvě dochází k rozdělení datového proudu do segmentů v případě odesílatele nebo k jejich opětovnému spojení v případě příjemce. Primárním úkolem této vrstvy je navázání spolehlivého či nespolehlivého spojení podle požadavků a charakteru služby, jež poskytuje data.

- **Síťová vrstva** – Síťová vrstva zajištuje adresaci paketů v síti. Pomocí směrovacích protokolů je vybírána nejvhodnější trasa pro přenos paketů k cílové stanici. Na této úrovni dochází k adresaci pomocí logických adres.

- **Linková vrstva** – Na této vrstvě jsou pakety ze síťové vrstvy zapouzdřeny do rámců a adresovány pomocí adres MAC. Také se na této vrstvě provádí kontrolní součty rámců a řízení přístupu k přenosovému médiu.

- **Fyzická vrstva** – Nejspodnější vrstva modelu slouží k převodu rámců na posloupnost bitů, které jsou v médiu šířeny nejčastěji jako elektrické či světelné signály. Stará se také o odesílání a příjem těchto signálů. Do této vrstvy spadají například i specifikace týkající se vlastností signálů dle charakteru přenosových médií.

Více o modelu ISO/OSI lze nalézt v (Lammle, 2010, s. 50−66).

2.2 Příklady útoků na jednotlivých vrstvách

Tabulka 1 – Příklady útoků na jednotlivých vrstvách modelu ISO/OSI

Vrstva	Možné útoky
Aplikační	viry, červi, trojské koně, boti, DNS spoofing, SQL injection, DDoS (HTTP, DNS, SMTP, VoIP), phishing útoky
Prezentační	SSL MITM, SSL DoS
Relační	DNS poisoning, Session hijacking, Telnet DDoS
Transportní	SYN flood, Teardrop, Man In The Middle, skenování portů
Síťová	ICMP flood, ICMP redirect, Ping of Death, IP spoofing
Linková	ARP spoofing, MAC flooding, ARP cache poisoning, Port scanning, MAC spoofing
Fyzická	přerušení/odpojení zdrojového nebo síťového kabelu, fyzické zničení síťového zařízení, rušení přenosu

3 Architektura TCP/IP

Architektura TCP/IP byla vyvinuta agenturou DARPA ministerstva obrany USA v 70. letech. Na základě této architektury a jejích technologií je dnes realizována většina sítí Internetu. TCP/IP v sobě zahrnuje sadu protokolů, jejichž cílem je zajištění spolehlivé a odolné komunikace.

3.1 Model TCP/IP

Čtyřvrstvý model určený k popisu komunikace na sítích používající tuto technologii. Jedná se o starší model než model ISO/OSI. Více o modelu TCP/IP lze nastudovat v (Peterka, 1992a; Osterloh, 2003, s. 23–27).

Obrázek 3.1 – Model TCP/IP

- **Procesní/aplikační vrstva** – Tato vrstva poskytuje protokoly pro aplikace, jež komunikují s ostatními uzly sítě. Odpovídá aplikační, prezentační a relační vrstvě modelu ISO/OSI. V případě potřeby tedy mohou být jejich služby realizovány samostatně na této vrstvě.

- **Transportní vrstva** – Transportní vrstva zajišťuje přenos dat mezi stanicemi. Nejčastěji je přenos realizován protokolem TCP[4] další možností je protokol UDP[5]. Stará se také o navázání spojení, zajištění integrity dat, umožnuje spojovaný či nespojovaný charakter přenosu.

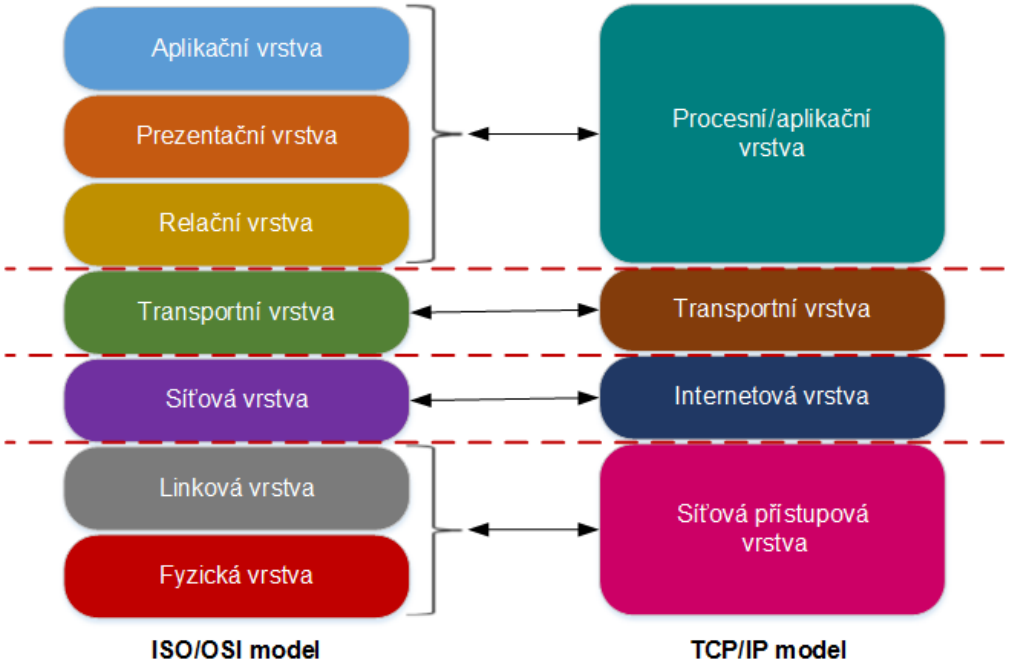
- **Internetová vrstva** – Princip fungování této vrstvy je analogický se síťovou vrstvou modelu ISO/OSI (vizte kapitola 2.1), tedy zajištění logického směrování přes jednotlivé směrovače.

[4] TCP – zajišťuje spojovaný (spolehlivý) přenos. Více v (Peterka, 1992b; Osterloh, 2003, s. 231–258).
[5] UDP – zajišťuje nespojovaný (nespolehlivý) přenos. Více v (Odvárka, 2001; Osterloh, 2003, s. 259–264).

- **Síťová přístupová vrstva** – Hlavním úkolem síťové přístupové vrstvy je adresování v sítí pomocí hardwarových adres MAC a přístup k fyzickému médiu. Ekvivalentní s touto vrstvou jsou linková a fyzická vrstva modelu ISO/OSI. Poskytuje protokoly pro fyzický přenos paketů v síti.

3.2 Porovnání modelů ISO/OSI a TCP/IP

Model ISO/OSI se využívá spíše k teoretickému popisu fungování přenosu dat sítí, právě kvůli jeho názornému rozčlenění vrstev (snazší pochopení funkcí na každé vrstvě, menší složitost sítě). Model TCP/IP je pak uplatňován při praktických realizacích sítí. Jak je patrné (Obrázek 3.2), mezi vrstvami obou modelů existuje vzájemná korelace.



Obrázek 3.2 – Porovnání modelů ISO/OSI a TCP/IP

3.3 Vybrané bezpečnostní protokoly architektury TCP/IP

Jak již bylo dříve poznamenáno, v době vzniku architektury TCP/IP potažmo jejich protokolů nebyl požadavek na bezpečnost důležitý jako dnes. Logickým vyústěním této situace tedy bylo dodatečně navrhnout protokoly, které problémy se zabezpečením řešily.

3.3.1 IPsec

S masovým rozšiřováním internetu postaveného na architektuře TCP/IP se stal protokol IP nejpoužívanějším celosvětovým protokolem pro komunikaci. To však také vedlo k tomu, že začaly na povrch vyplouvat chyby a nedostatky tohoto protokolu. Jedním z nich, byla nemožnost zajistit bezpečnou komunikaci (bezpečný přenos dat mezi dvěma uzly). Proto se začal vytvářet nový komunikační protokol IPv6 s již implementovanými bezpečnostními prvky (IPsec je tedy jeho nutnou součástí). Avšak současně s tím bylo třeba reagovat i na skutečnost, kdy současný protokol IP ve verzi 4 byl a stále je ve světě internetu hojně využíván. Došlo tak k začlenění volitelné sady protokolů IPsec do protokolu IPv4. IPsec

poskytuje základní tři vlastnosti: integritu[6], autenticitu[7] a utajení (šifrování) IP paketů[8]. Tento protokol se nachází na internetové vrstvě modelu TCP/IP.

Jak již bylo výše napsáno protokol IPsec se skládá ze sady bezpečnostních protokolů (AH, ESP) a protokolu pro správu šifrovacích klíčů (IKE). IPsec rovněž zahrnuje koncept bezpečnostní asociace (SA), jak popisuje Pužmanová (2006, s. 381–384).

- **Bezpečnostní asociace** (**SA**) – Udává, jakými bezpečnostními opatřeními (bezpečnostní protokol, algoritmus šifrování, typ přenosu, klíč) se budou vzájemně dvě entity při své komunikaci řídit. Asociace je jednosměrná, takže je třeba pro každou entitu provést samostatnou asociaci a to ještě před začátkem samotného procesu předávání dat. Každá asociace je identifikována pomocí jednoznačného identifikátoru (SPI). V případě nového spojení přes IPsec musí uzel založit novou SA. Každý uzel při používání IPsec v sobě uchovává dvě databáze:
  - **Databázi bezpečnostních asociací** (SPA) – Pro uchovávání bezpečnostních opatření pro každou SA.
  - **Databázi bezpečnostní politiky** (SPD) – Obsahuje seznam bezpečnostních politik, které se mají uplatnit na jednotlivé datagramy v závislosti na selektoru (použít IPsec, zahodit datagram, nepoužívat IPsec).

Popis základních protokolů IPsec:

- **IKE** – Pomocí tohoto protokolu se zprostředkuje spojení IPsec. Ještě předtím však musí zkontrolovat autenticitu uzlu, se kterým má být bezpečná komunikace navázána.
- **AH** – Protokol, jenž se stará o zajištění autenticity a integrity přenášeného paketu. Do datagramu IP je přidána hlavička AH, vizte Tabulka 2. AH protokol neumožnuje šifrování paketu.

[6] Paket nebyl během přenosu změněn.
[7] Odesílatel se dozví, od koho byl paket poslán.
[8] Pokud útočník odchytí paket, nedozví se informaci, kterou přenáší.

Tabulka 2 – Struktura hlavičky protokolu AH

8 bitů	8 bitů	16 bitů
Další hlavička (vyššího protokolu)	Délka dat	Rezervováno
Index bezpečnostního parametru (SPI)		
Sekvenční číslo		
Autentizační informace		

šířka 32 bitů

- **ESP** – Protokol ESP se stará o autenticitu, integritu přenášených dat a navíc i o zašifrování přenášeného paketu například pomocí algoritmu DES, 3DES nebo IDEA. Hlavička protokolu ESP, vizte Tabulka 3, je opět přidána do paketu IP.

Tabulka 3 – Struktura hlavičky protokolu ESP

Index bezpečnostního parametru (SPI)		
Sekvenční číslo		
Data		
Doplnění		
	Délka doplnění	Další hlavička (vyššího protokolu)
Autentizační informace		

šířka 32 bitů

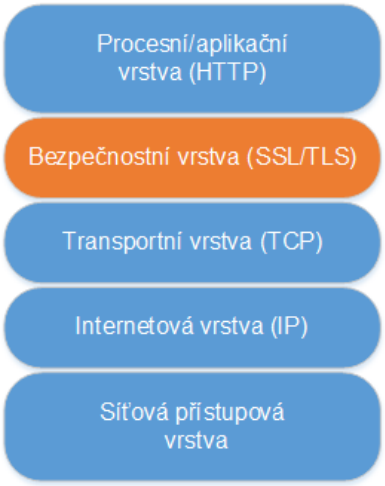
Více informací k tomuto tématu lze nastudovat v (Northcutt et al., 2005, s. 190–195).

3.3.2 SSL/TLS

Protokol SSL vznikl opět důsledkem expanze Internetu a s ním spojené využití ve většině sfér lidských činností, kdy bylo třeba čelit hrozbám ohledně nezabezpečeného přenosu dat, například jejich odposlouchávání a falšování.

O vývoj protokolu se zasloužila firma Netscape, jelikož však nebyl protokol proprietární, podíleli se na vývoji i další vývojáři. V roce 1996 došlo standardizační organizací Internet Engineering Task Force k přejmenování protokolu SSL na TLS (verze SSL 3.0 a TSL 1.0 jsou až na drobné detaily prakticky shodné). Z toho důvodu, jelikož je zkratka SSL známější a ustálenější, bude dále v textu upřednostněna.

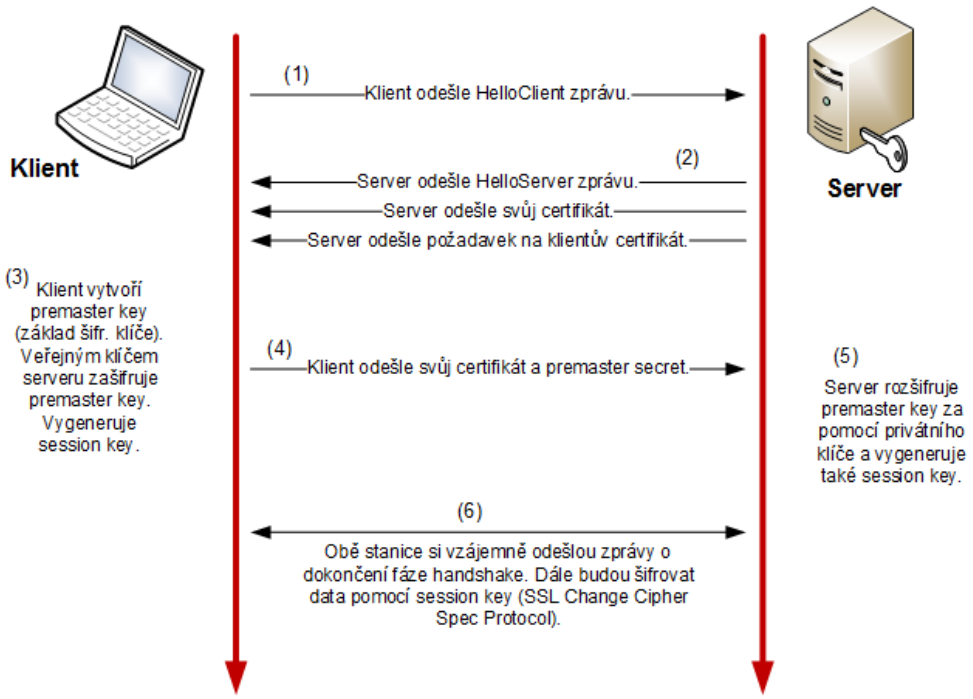
Z názvu SSL vyplývá, jedná se o bezpečnostní vrstvu vkládanou mezi aplikační a transportní vrstvu modelu TCP/IP, jak znázorňuje Obrázek 3.3. Protokol zajišťuje integritu a šifrování dat z aplikační vrstvy a také autentičnost odesílatele při přenosu. Pro přenos je třeba uplatnit spolehlivou (spojovanou) komunikaci za pomocí protokolu TCP. K šifrování dat se používá symetrické šifrování, asymetrické šifrování (pomocí veřejného a privátního klíče) je použito k ověření komunikujících uzlů a přenosu symetrických klíčů.



Obrázek 3.3 – Model TCP/IP s bezpečnostní vrstvou SSL/TLS

SSL se skládá ze 4 protokolů:

- **SSL Handshake Protocol** – Úkolem protokolu je navázání bezpečného komunikačního spojení mezi dvěma entitami. Tento úkol tak představuje několik činností (Obrázek 3.4).



Obrázek 3.4 – Navázání šifrované komunikace SSL Handshake protokolem

- **SSL Record Protocol** – Protokol zajišťuje zabalení (encapsulation) dat a přidání hlavičky. Takto vzniklý objekt je pak nazýván record. Celý proces je rozdělen do 5 fází (shrnuje Obrázek 3.5):

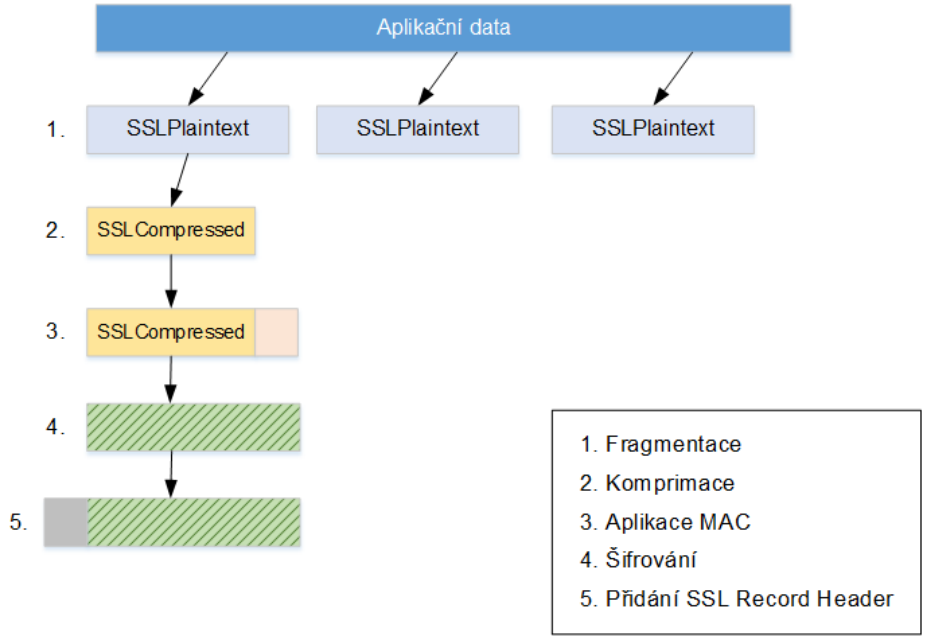
    1. **Fragmentace** – Data jsou zde rozdělena do samostatných textových SSL recordů o maximální délce 2^14 bytů. Fragmentovaná data jsou pak označována jako SSLPlaintext.

    2. **Komprimace** – Na SSLPlaintext je aplikována komprimační bezztrátová metoda, jež byla dohodnuta v handshake procesu, vizte *SSL Handshake Protocol*. Na konci komprimace je SSLPlaintext zapouzdřen do tzv. SSLCompressed record data.

    3. **Aplikace MAC**[9] – V této fázi dochází k výpočtu MAC za pomocí hash funkce (MD5 nebo SHA) opět předem dohodnuté v handshake procesu, vizte *SSL Handshake Protocol*. Kód MAC je poté přidán k SSLCompressed record data.

    4. **Šifrování** – Pro zašifrování SSLCompressed record data je využito proudového[10] nebo blokového[11] šifrovacího algoritmu. Výstupní velikost zašifrovaných record data nesmí překročit velikost 2^14 bytů.

    5. **Přidání SSL Record Header –** K výsledným record data je přidána hlavička, čímž vzniká objekt record.



Obrázek 3.5 – Diagram fází SSL Record Protocol

[9] Message Authentication Code
[10] Record je šifrován tak, jak byl předán předchozím procesem.
[11] Record musí být před zašifrováním doplněn na požadovanou délku násobku bloku.

- **SSL Change Cipher Spec Protocol** – Tento protokol je aplikován v poslední fázi procesu handshake, vizte *SSL Handshake Protocol*. Během jeho činnosti dochází k přepnutí dvou entit z vyčkávacího do provozního režimu. Entity tak zahájí šifrovanou komunikaci.
- **SSL Alert Protocol** – SSL Alert Protocol ohlašuje chyby, které vznikly během spojení.

Bližší informace je možné nalézt v (Odvárka 2002a; 2002b; 2002c).

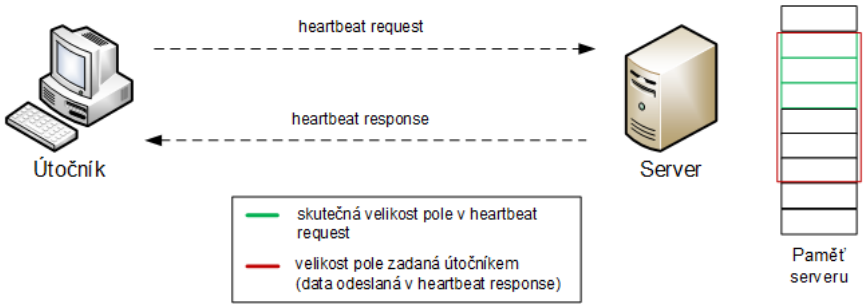
Chyba krvácejícího srdce (Heartbleed bug)

Jedná se o nově objevenou chybu, která byla zveřejněna v dubnu letošního roku. Týká se knihovny OpenSSL, která v sobě implementuje protokol TLS (dříve SSL). Jelikož tato chyba představuje jedno z největších bezpečnostních ohrožení na Internetu, jenž bylo dosud zaznamenáno v souvislosti se zabezpečeným spojením, je proto nezbytné se v této práci o ní zmínit a popsat princip útoku těžícího z této chyby.

Masivní rozsah bezpečnostního rizika je především dán tím, že knihovnu OpenSSL využívá většina serverů. Při šifrované komunikaci, zejména pak u nespojovaných protokolů, jako je UDP, se relace udržuje pomocí tzv. „heartbeatingu“. Jedná se o zasílání *heartbeat requests*, na které komunikační partner odpovídá *heartbeat responses*. Podstata Chyby krvácejícího srdce spočívá v délce pole *payload* u *heartbeat requests*, přesněji při zadávání hodnoty délky pole se neověřovala skutečná velikost pole.

V případě, kdy útočník zadá do pole hodnotu větší než je skutečná velikost pole a odesílá na server takto upravené *heartbeat requests*, server tyto požadavky nejprve uloží do své paměti a následně zkopíruje do *heartbeat responses* takové množství dat z paměti, které bylo v *heartbeat requests* deklarováno. Tím byla překopírována i část paměti, jenž bezprostředně následovala za těmito daty. V paměti serveru se nacházejí přístupové údaje, a dokonce i privátní klíč. Útočník tak může všechny tyto údaje získat. (Caletka, 2014)

Chyba se v knihovně vyskytovala více než 2 roky, mohla tedy být zneužita ve velkém měřítku. Všem uživatelům je doporučeno změnit si svá přístupová hesla u důležitých účtů (zejména u serverů, které uchovávají data s finančním charakterem). Pro servery je doporučeno aktualizovat OpenSSL na nejnovější verzi, kde je již chyba opravena, dále si nechat vystavit nové certifikáty a revokovat původní, potenciálně kompromitované certifikáty.



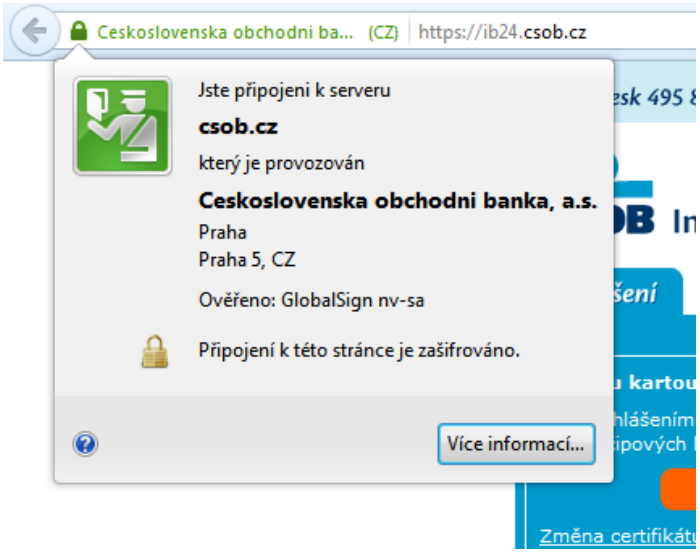
Obrázek 3.6 – Schéma útoku využívajícího Chyby krvácejícího srdce

3.3.3 HTTPS

Protokol HTTPS, neboli Hypertext Transfer Protocol Secure, představuje bezpečnostní nástavbu protokolu HTTP, jak popisuje Sosinsky (2010, 676–678). HTTPS se skládá ze samotného protokolu HTTP a šifrovacího protokolu SSL/TLS, který byl představen v kapitole 3.3.2. Standardně protokol využívá port 443 na straně serveru.

HTTPS se používá v případech, kdy je třeba chránit data přenášená po nechráněné sítí (nejčastěji Internetu) před odposloucháváním. Data jsou zašifrována pomocí protokolu SSL/TLS a následně přenesena za pomoci protokolu HTTP. Aby však mohlo být šifrování provedeno, musí dojít k výměně veřejných klíčů mezi serverem a klientem. Protokol totiž využívá asymetrického šifrování.

Dalším úkolem protokolu může být také zajištění autentifikace (ověření identity) serveru. K tomu se nejčastěji využívají digitální certifikáty, jenž jsou vytvářeny certifikační autoritou. Autentifikace tak slouží k obraně před podvrhnutím dat při útoky typu Man in the middle.



Obrázek 3.7 – Informace o digitálním certifikátu domény csob.cz (Mozilla Firefox 28.0)

4 Prevence a obrana proti síťovým útokům

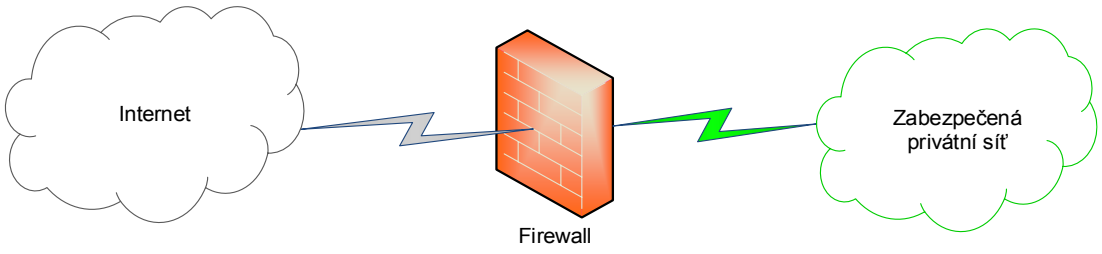
Otázka bezpečnosti sítě obecně, tak, aby byla úroveň zabezpečení co možná nejúčinnější proti dnes nejčastějším útokům, není pro správce sítě jednoduchou záležitostí. Je třeba vzít v úvahu spoustu faktorů. Nejlepším způsobem, jak mít síť, co možná nejlépe chráněnou, je zavedení hloubkové obrany. Pomocí této obrany lze vytvořit jednotlivé bezpečnostní vrstvy sítě a zajistit tak její obranné zapouzdření. Výhodou tohoto zapouzdření je, že v případě prolomení obranného prvku jedné vrstvy, zůstávají aktivní další obranné prvky v ostatních vrstvách. Útočník by tak musel prolomit více vrstev, aby dosáhl úplného průniku.

Hloubkovou obranu tedy zajišťují různé bezpečnostní prvky a opatření, které budou popsány dále v této kapitole. Podrobněji se problematikou bezpečnosti síťové infrastruktury zabývají Northcutt et al. (2005, s. 11–171).

4.1 Firewally

Též někdy označovány pojmem bezpečnostní brány, jsou síťová zařízení, jejichž úkolem je zabezpečení a řízení komunikace (datového toku) mezi různě důvěryhodnými sítěmi. Postupným vývojem vznikaly různé druhy firewallů (popsaných dále), specializující se na zkoumání dat z odlišných pohledů.

Přestože v dnešní době existují různé druhy firewallů, jejich základní princip fungování je založen na sadě pravidel, pomocí nichž se rozhoduje o provedení příslušné akce s pakety, procházejícími skrze firewall. Mohou tak být propuštěny dále případně zablokovány. Z toho tedy vyplývá nutnost, aby byla požadovaná síť chráněna, musí veškerá komunikace mezi chráněnou sítí a ostatními sítěmi procházet právě přes toto zařízení. Filtrování komunikace může probíhat na druhé až sedmé vrstvě modelu ISO/OSI. Firewally lze implementovat softwarově nebo jako samotná hardwarová zařízení. Jednotlivé druhy firewallů jsou rozepsány níže.



Obrázek 4.1 – Schéma komunikace mezi sítěmi skrze firewall

4.1.1 Paketový filtr

Z historického pohledu se jedná o nejstarší druh firewallu. Z pohledu modelu ISO/OSI pracuje tento na úrovni transportní a síťové vrstvy. Princip fungování je založen na zkoumání adresy IP (zdrojové nebo cílové) přesněji její hlavičky a podle předem daných pravidel filtr rozhodne, zda příchozí paket propustí do sítě či jej odfiltruje (zahodí). Analogicky může filtrovat i odchozí komunikaci. Pro detailnější řízení komunikace například služeb musí filtr

prozkoumat informace přenášené v rámci transportní vrstvy, a to čísla portů, jak popisuje Peterka (2001). Na základě tohoto prozkoumání pak lze filtrovat komunikaci pouze určitých služeb. Díky tomu tak poštovní server přijímá a odesílá pouze elektronickou poštu, podobně lze tato pravidla stanovit i pro ostatní servery jako FTP, HTTP atd.

Nevýhodou paketového filtru je fakt, že nedokáže prozkoumat filtrované pakety do mnohem větší hloubky, tedy až do úrovně aplikační vrstvy. V důsledku toho tak útočník může data zapouzdřená ve vyšších vrstvách využít k útokům, které nebudou pomocí paketového filtru detekovány. Na druhou stranu však paketový filtr představuje poměrně rychlý firewall (díky tomu, že prozkoumává komunikaci do úrovně transportní vrstvy), který může své uplatnění naleznout uvnitř hraničního routeru, kde může odrazit jednodušší formy útoků, se kterými se pak nemusí zabývat ostatní bezpečnostní prvky uvnitř sítě.

Příkladem paketového filtru na softwarové úrovni je Cisco ACL.

- **Cisco ACL (Access Control List)** – Řešení paketového filtru od společnosti Cisco. Filtruje pakety pomocí přístupových seznamů (ACL). Dělí se podle typu filtrování na standardní, rozšířený a reflexivní.

    a) **Standardní ACL** – Filtruje komunikaci na základě zdrojových adres. Základní syntaxe tohoto přístupového listu v Cisco IOS:

```
Router(config)# access-list <číslo seznamu 1-99> <permit | deny>
<zdrojová adresa> <maska> {log}
Router(config-if)# access-group <číslo seznamu> <in | out>
```

    b) **Rozšířený ACL** – Doplňuje standardní ACL o možnosti filtrování pomocí dalších kritérií například cílové adresy, typu protokolu, čísla portu atd. Základní syntaxe:

```
Router(config)# access-list <číslo seznamu 100-199> <permit | deny>
<protokol> <zdrojová adresa> <zdrojová maska> <zdrojový port>
<cílová adresa> <cílová maska> <cílový port> {log | log-input}
Router(config-if)# access-group <číslo seznamu> <in | out>
```

    c) **Reflexivní ACL** – Tento typ zajišťuje dynamické filtrování. Jednotlivé filtry se u každého spojení (relace) vytvářejí automaticky a po jeho ukončení jsou smazány. Jde o podmnožinu rozšířeného ACL – pojmenovaný přístupový seznam. Jeho definice se skládá z příkazů definování názvu a definování permit, deny, jak ukazuje základní syntaxe:

```
Router(config)# ip access-list extended <název seznamu>
Router(config-ext-nacl)# permit <protokol> {<zdrojová adresa>
<zdrojová maska>} {any} {<cílová adresa> <cílová maska>} {any} {eq
<port>} reflect <název seznamu>
Router(config-ext-nacl)# deny <protokol> {<zdrojová adresa>
<zdrojová maska>} {any} {<cílová adresa> <cílová maska>} {any} {eq
<port>} reflect <název seznamu>
```

```
Router(config-if)# ip access-group <název seznamu> <in | out>
```

Northcutt et al. (2005, s. 29–54)

4.1.2 Stavový paketový filtr

Tento typ firewallu opět pracuje s pakety na třetí a zejména čtvrté vrstvě modelu ISO/OSI. Ovšem, na rozdíl od bezstavového paketového filtru, při filtrování paketů sleduje informace o síťovém spojení tzv. stav paketu. Pomocí těchto informací se pak rozhoduje, zda je paket již součástí probíhající relace, pokud ano, mohou pakety procházet bez podrobnější inspekce firewallem. Pokud není předchozí podmínka splněna, dochází k procesu rozhodování, na základě sady pravidel o založení nové relace nebo případného blokování paketu.

Pro uchovávání informací o jednotlivých relací (spojení) v sobě firewall zahrnuje stavovou tabulku. V tabulce jsou pak ukládány hlavní atributy o každé relaci, především zdrojová adresa IP, cílová adresa IP, čísla portů a sekvenční čísla paketů, která se během komunikace dynamicky mění. Každý záznam (relace) je v tabulce uchováván pouze po určitou dobu. Po skončení relace je záznam vymazán.

Při vytváření nové relace se proces filtrování řídí podle typu spojení, které máme u architektury TCP/IP dvojího typu:

- **Spojované (pomocí protokolu TCP)** – U tohoto spojení firewall zkoumá, jestli paket obsahuje příznak SYN, který je u protokolu TCP typický pro založení spojení, jde o tzv. *Three Way Handshake proces*. Pokud tedy paket obsahuje tento příznak a splňuje další pravidla, je vytvořena nová relace a záznam přidán do stavové tabulky. Dalším paketům patřící do této relace je umožněn průchod, opět však musejí obsahovat správné parametry (např. očekávané sekvenční číslo). Spojení se ukončuje pomocí příznaku FIN v ukončovacím paketu. Následně dojde k ukončení celé relace a tím i odstranění záznamu ze stavové tabulky. Pro všechny případy stavový filtr obsahuje časovač, který umožňuje ukončit relaci a odstranit záznam z tabulky v případě nekorektního ukončení spojení.
- **Nespojované (řízené protokolem UDP)** – Jelikož se jedná o nespojovou komunikaci, nelze u protokolu UDP stav uvažovat. Lze však spojení sledovat pseudo-spojovou metodou, kdy jsou zkoumány určité znaky tohoto spojení. Pakety také zároveň neobsahují sekvenční čísla ani příznaky, takže zkoumání stavu spojení je odkázáno na adresy IP a čísla portů. Pakety také neobsahují příznak FIN pro oznámení ukončení spojení jako u TCP. Z toho důvodu tak stavový paketový filtr ukončí relaci a vymaže záznam ze stavové tabulky po uplynutí stanoveného časového limitu, během něhož nedojde k žádným aktivitám v relaci.

Jako výhodu paketového filtru lze uvést rychlé filtrování, jenž je docíleno kontrolou paketů na třetí a čtvrté vrstvě a také jemnější kontrolou paketů patřících do probíhající relace. Slabinou

tohoto typu firewallu pak mohou být různé útoky typu DoS, zejména pak útok SYN flood, vizte kapitola 6.3.3.

Příklady firewallů, spadající do této kategorie jsou:

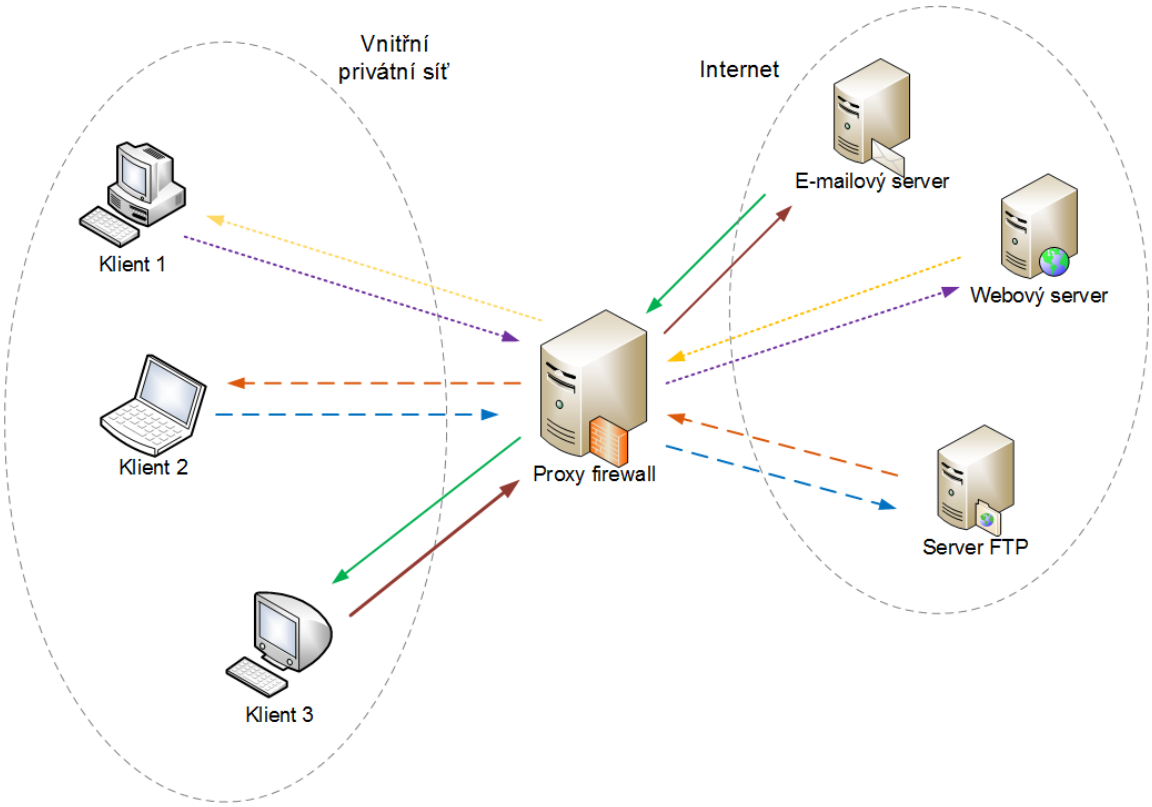
- **iptables** – Lze zařadit mezi stavový paketový filtr určený pro linuxové a unixové systémy. Pomocí tohoto nástroje však lze vytvořit různé druhy firewallů.
- **Cisco ACL** (**Access Control List**) – Mezi stavový paketový filtr patří také ACL, konkrétně typ reflexivní ACL, který umožňuje sledování spojení pomocí protokolů TCP nebo UDP a vytváření dynamických přístupových seznamů.

4.1.3 Proxy firewall

Proxy firewall též nazývána aplikační brána. Northcutt et al. (2005, s. 81–87) v knize uvádí, že se jedná o druh firewallu, provádějící analýzu paketů (přesněji aplikačních dat) z hlediska modelu ISO/OSI na aplikační vrstvě. Díky tomu, že dokáží zajistit ochranu na všech vrstvách, řadí se mezi neúčinnější a nejbezpečnější druh firewallu. Proxy firewall při své činnosti zaujímá roli prostředníka, jenž zprostředkovává vzájemnou komunikaci mezi klientem a serverem (tento princip je popsán dále). V případě klienta se firewall chová jako server (posluchač). Pro server, se kterým požaduje klient skutečné spojení, představuje firewall klienta (iniciátora). Server ani klient tak nezjistí, s kým je vedena skutečná komunikace právě díky zprostředkované komunikace přes proxy firewall. Pro každý protokol aplikační vrstvy je třeba mít samostatnou proxy službu a nad ní prováděnou kontrolu, případně je možné použít obecný proxy server.

Princip komunikace prostřednictvím aplikační brány

Příklad komunikace (Obrázek 4.2): Klient 1 má v úmyslu zobrazit webovou stránku. Naváže tak spojení s proxy serverem, který v tuto chvíli zastává roli posluchače. Po příslušných kontrolách se stává iniciátorem a navazuje komunikaci s webovým serverem. V dalším kroku webový server zasílá proxy serveru zpět odpověď, opět dochází ke kontrolám a pokud jsou vyhodnoceny pozitivně, zasílá proxy server odpověď klientovi 1.



Obrázek 4.2 – Schéma komunikace prostřednictvím proxy firewallu

K hlavním výhodám tohoto firewallu patří zejména:

- Silná ochrana, díky hloubkové analýze komunikace (až do aplikační vrstvy).
- Skrytí adres IP ve vnitřní síti, čímž odpadají problémy s falšováním těchto adres.
- Generování záznamů ze služeb proxy o narušení zásad zabezpečení firewallu.
- Skrytí topologie chráněné sítě.

Ani proxy firewall však nepředstavuje dokonalé řešení zabezpečení síťové infrastruktury, mezi jeho nevýhody patří:

- Pomalejší zpracování komunikace (analýzy dat) z důvodu prověření všech vrstev síťového modelu TCP/IP.
- Složitější konfigurace proxy firewallu oproti paketovému filtru. Zejména pak v případě nastavení služby pro procházení všech aplikací.
- Omezený počet služeb proxy provozovaných v rámci firewallu. Pro další protokol/aplikaci komunikující přes firewall je třeba vytvořit novou proxy službu.
- Zranitelnost operačního systému, na kterém je firewall provozován. Operační systém se může stát terčem útoků, což by mělo neblahý vliv na fungování samotného firewallu. Dále pak fungování firewallu můžou ovlivnit i případně chyby operačního systému.

Příkladem proxy firewallů jsou nástroje SOCKS, iptables.

4.2 Systém pro odhalení průniku (IDS)

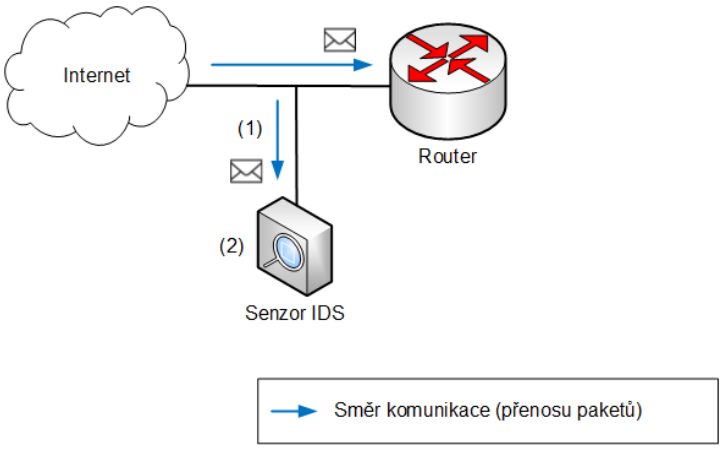
Systém pro odhalení průniku představuje další ochranný prvek pro zabezpečení síťové infrastruktury. Definuje tak další úroveň zabezpečení. Kontroluje síťový přenos a pomocí definovaných signatur hledá případné bezpečnostní hrozby jako skenování sítě nebo samotné útoky.

Na rozdíl od firewallu nemá IDS v případě detekování možnost zabránit průniku útočníka do sítě, ale v případě nalezení podezřelých aktivit provádí jejich logování a zasílá správci výstrahy o narušení. Též může v případě zjištění útoku překonfigurovat firewall. Nastavení a definování signatur, tak aby byly co nejvíce efektivní, tedy s co nejmenším počtem falešných detekcí a negativních zhodnocení[12], představuje značně složitý úkol a vyžaduje tak zkušeného správce, který rozumí problematice tohoto systému a fungování dané sítě.

Northcutt et al. (2005, s. 149–171) rozdělují systém pro odhalení průniku z hlediska funkcionality na tyto části:

4.2.1 Senzory

Senzory jsou prvky IDS, jde o pasivní sondy, které jsou v závislosti na konkrétních vlastnostech, specifikacích sítě a především podle jejich počtu (který určují finanční možnosti společnosti) rozmístěny v určitých segmentech síťové infrastruktury. Představují stěžejní část činnosti celého systému, a to analýzu komunikace a zasílání výstražných zpráv. Princip fungování senzoru znázorňuje Obrázek 4.3. Komunikace je kopírována na port senzoru (1). Senzor následně analyzuje data z jednotlivých paketů a snaží se najít řetězce, shodující se se signaturami (2). Pokud taková shoda nastane, senzor vygeneruje výstrahu do systému IDS.



Obrázek 4.3 – Schéma principu fungování senzoru IDS

4.2.2 Systém

Zajišťuje v síti, kde je instalován IDS, centrální bod. Systému jsou ohlašovány výstražné zprávy, archivuje výstrahy do databáze, zajišťuje komunikaci se senzory. Slouží jako prostředí pro správce sítě, obvykle se jedná o soubor softwarových nástrojů.

[12] Situace, kdy při výskytu bezpečnostní hrozby nedojde k vygenerování výstrahy.

4.2.3 Signatury

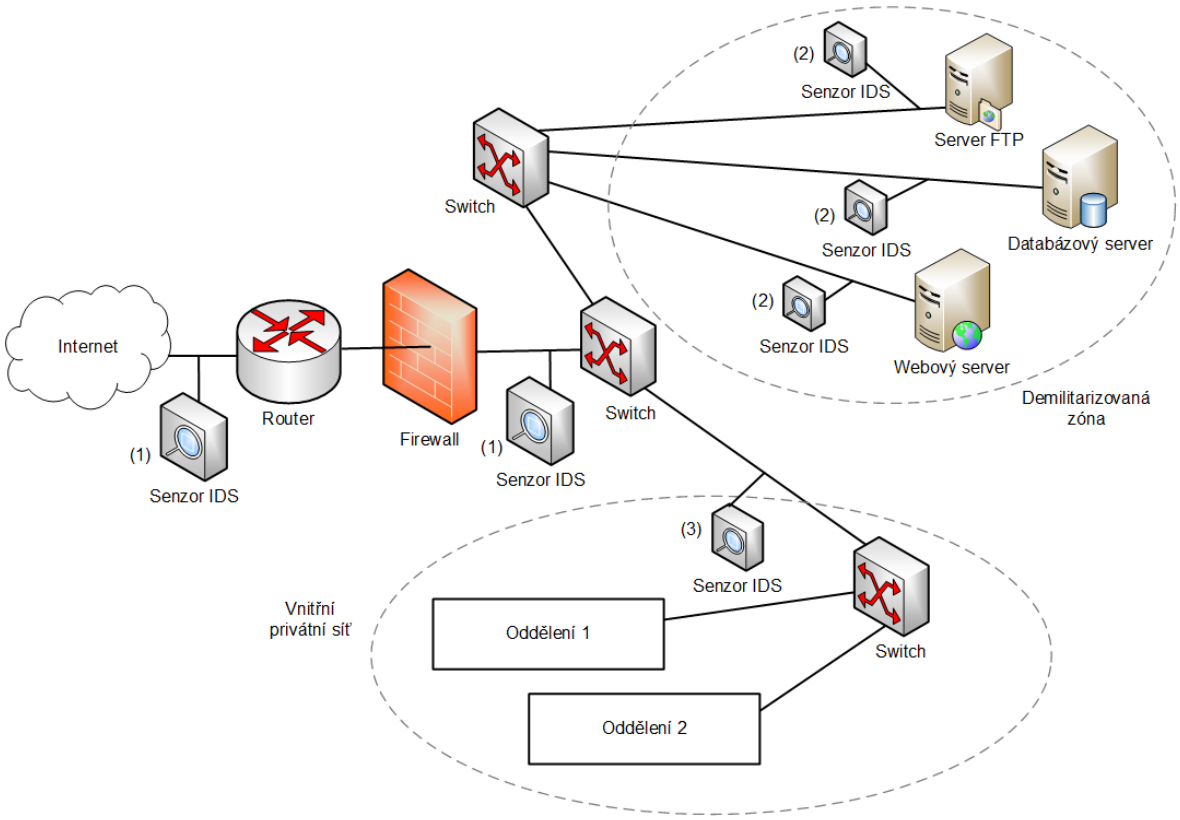
Představují vzory útoků nebo podezřelých aktivit, které jsou uchovávány v databázi IDS. Při analýze síťového přenosu dochází k porovnávání s těmito signaturami a v případě shody jsou generovány výstrahy. Vytváření signatur skýtá úskalí, jelikož příliš obecně definovaná signatura může generovat velké množství falešných detekcí průniku. Naopak specifičtější definice vede k negativnímu zhodnocení průniku.

U distribuovaných systémů IDS, které jsou nejhojněji využívány, administrátoři sítí posílají společnostem poskytující tyto systémy soubory s výstrahami (logy) ze svých senzorů IDS. Společnosti poté výstrahy analyzují a vytváří pro své zákazníky aktualizace s nově přidanými signaturami podobně jako je známo u antivirových programů.

4.2.4 Umístění senzorů IDS v síti

Otázka volby počtu senzorů je závislá na finančních možnostech organizace. Obecně platí, že pro efektivní zabezpečení by mělo být využito většího počtu senzorů IDS. V praxi je obvyklé použití většího počtu senzorů u velkých společností a menšího počtu u malých firem. Dalším rozhodovacím faktorem při volbě počtu senzorů je také hodnota (cennost) informací, které mají být ochráněny. Umístění senzoru je tak závislé na struktuře sítě, její velikosti a počtu senzorů, která jsou k dispozici.

- **Větší počet senzorů** – Pokud organizace má možnost pořídit větší počet senzorů, znamená to lepší možnosti detekování útoků a vyšší schopnost obrany. Obrázek 4.4 ukazuje, jak je vhodné v takovém případě rozmístit senzory (1) před i za firewall. Vhodné je umístit senzor i do jednotlivých segmentů sítě (3) a nastavit každý podle typů přenosů, služeb a protokolů typických pro daný segment. Nejlepší možnost pak představuje umístění striktně nastaveného senzoru pro konkrétní službu (2).
- **Menší počet senzorů** – Malé firmy mají omezené finance, a tak si většinou mohou dovolit menší počet senzorů. Poté je tedy vhodné umístit senzory na místa, kterými prochází velké množství přenosů, nejčastěji na hranici sítě. Pokud by byl k dispozici pouze jeden senzor je z hlediska efektivity nejvhodnější umístit jej před nebo za firewall (1) – Obrázek 4.4. Nastavení senzoru musí být obecné a značně omezené tak, aby detekce zahrnovala co možná nejvíce typů komunikace. Před firewallem směrem do internetu, tak může zachytit většinu útoků a pokusů o proniknutí směřovaných do sítě. Za firewallem je možné zachytit útoky, které firewall nedokázal zablokovat.



Obrázek 4.4 – Možnosti umístění senzorů IDS v síti organizace

Zástupci systémů pro odhalení průniku jsou například Cisco Secure, Snort, Shadow.

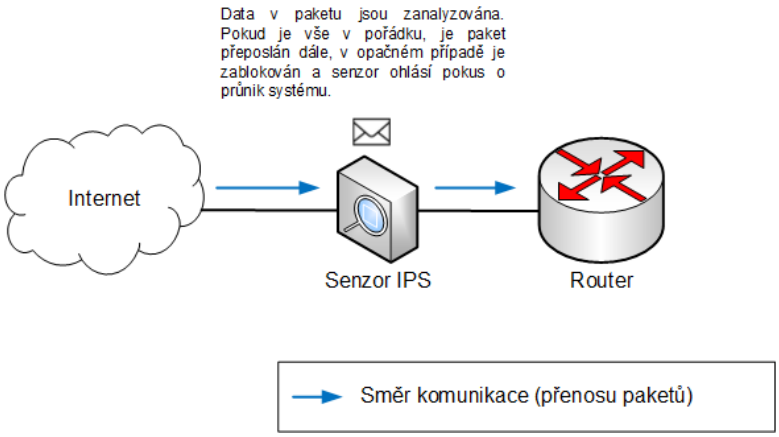
4.3 Systém prevence průniku (IPS)

Pasivní charakter senzorů IDS zajišťuje především detekci průniku. Má tak pouze omezené možnosti, jak procesům narušení zabránit např. rekonfigurováním firewallu. Neumožňuje však zastavit útok ještě před jeho uskutečněním. Tento problém proto řeší systém prevence průniku (IPS).

Hlavním rozdílem systému IPS oproti IDS je koncepce metod pro detekci narušení a implementace aktivní obrany. Senzory IPS jsou umístěny přímo v komunikační cestě (ukazuje Obrázek 4.5), takže při detekci útoku zahájí okamžitou protiakci (blokování přenosu). IPS je tak evolučním vylepšením IDS. Doplňuje nedostatky firewallu a vytváří další obrannou vrstvu.

Systém IPS se dělí na dva typy:

- **Uživatelské** – Jsou instalovány na uživatelských stanicích a chrání tak pouze jeden konkrétní uzel.
- **Síťové** – Jejich senzory jsou instalovány v rámci síti a mají tak podle konkrétního umístění v síti za úkol analyzovat komunikaci v rámci celé lokální sítě či některé její části (segmentu). Umístění senzorů pak odpovídá stejný pravidlům, popsaným v kapitole 4.2.4. (Černý, 2010, s. 22–25)



Obrázek 4.5 – Schéma principu fungování senzoru IPS

5 Simulační prostředí a nástroje pro simulaci

Cílem kapitoly je představení operačních systémů a především nástrojů použitých pro simulaci útoků.

5.1 Ubuntu 13.10 Saucy Salamander

Ubuntu je open source software[13] linuxová distribuce, šířená pod licencí GNU/GPL. Jeho využití je velmi rozmanité. Lze jej provozovat na různých hardwarových platformách, jako jsou cloud systémy, servery, osobní počítače, mobilní telefony, tablety. (Canonical Ltd., © 2014)

Operační systém byl využit pro nainstalování nástroje Imalse, s jehož pomocí byla následně provedena simulace útoků, vizte kapitola 6.3.1 a 6.3.2.

5.2 Microsoft Windows 7 Professional

Operační systém vyvinutý společností Microsoft. Jelikož se jedná v dnešní době o nejrozšířenější a mezi běžnými uživateli oblíbený systém, bylo jej využito pro instalaci nástroje VirtualBox a také sloužil pro některé simulované útoky, jakožto operační systém oběti.

5.3 Kali Linux

Kali Linux je linuxová distribuce pro pokročilé penetrační testování a bezpečnostní audit. Jedná se o nástupce známé distribuce BackTrack Linux, jenž byla kompletně přepracována a přejmenována na Kali Linux. Systém je na rozdíl od jeho předchůdce založen na distribuci Debian a jejích standardech. K dispozici je více než 300 nástrojů pro penetrační testování. (Offensive Security Ltd., 2013)

Distribuce obsahuje nástroje pro MITM útoky, DoS útoky, analýzu paketů, skenování portů, Wi-Fi cracking, útoky za pomoci sociálního inženýrství, password cracking atd.

5.4 Oracle VM VirtualBox

Oracle VM VirtualBox – jedná se o nástroj pro virtualizaci operačních systémů. Vyvíjený jako open source software pod GNU/GPL licencí. Podporuje všechny dnes nejdostupnější operační systémy: Windows, Linux, Macintosh a Solaris. (Oracle Corporation, © 2004–2014)

VirtualBox byl nainstalován na Windows 7 Professional. Dále došlo k virtualizaci distribuce Kali Linux prostřednictvím tohoto nástroje.

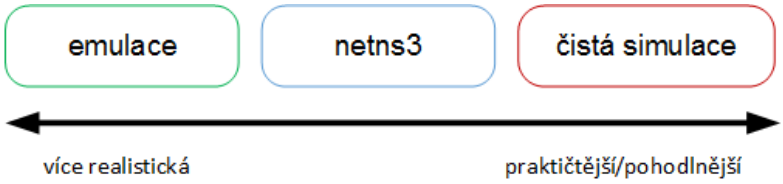
[13] Otevřený software, u něhož je volně dostupný zdrojový kód. Lze jej volně využívat a modifikovat při dodržení stanovených podmínek.

5.5 Imalse

Představuje projekt, který vznikl v rámci *Google Summer of Code 2012*[14]. Vytvořil jej Jing Conan Wang, student doktorského programu na Bostonské univerzitě.

Imalse neboli Integrated Malware Simulator and Emulator je framework, jak popisuje Wang (2012), umožňující vytvořit fiktivní botnet stanic infikovaných malwarem. Framework umožňuje nastavit (implementovat) jakou činnost má malware, jenž ovládá infikovanou stanici, provádět. Imalse provádí simulace v jednom z těchto tří režimů:

- **Režim emulace** – Při zvolení tohoto režimu se budou jednotlivé instance Imalse chovat jako skutečný malware. Lze jej tak nainstalovat na reálnou nebo virtuální stanici a testovat charakteristiku malware.
- **Netns3 simulační režim** – Umožňuje navrhnout topologii simulované sítě a specifikovat adresy IP jednotlivým uzlům. Zahájením simulace Imalse spustí virtuální hosty (linux namespace) pro každý definovaný uzel a automaticky vytvoří simulační síť. Všechny virtualizované uzly se připojí k simulátoru ns-3 a veškerý provoz tak bude probíhat v rámci simulátoru. Simulace probíhá v reálném čase.
- **Ns-3 čistý simulační režim** – Celá simulace probíhá v rámci simulátoru ns-3, nejsou spouštěny virtuální hosté. Namísto reálného času simulace je použit plánovač NS3. Jeden simulační den tak může trvat pouze několik sekund.



Obrázek 5.1 – Režimy simulace v Imalse

Framework ke své činnosti využívá další nástroje:

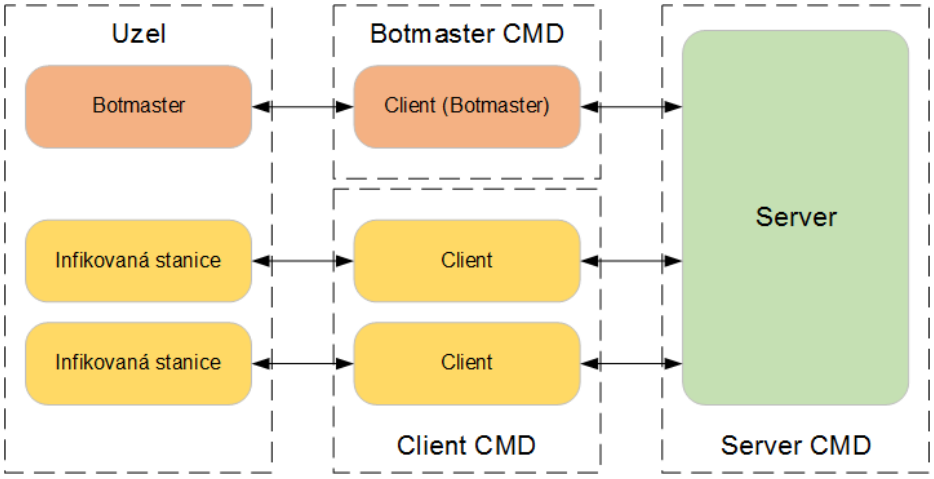
- **Ns-3** – Síťový simulátor založený na diskrétních událostech, určený pro vědecké a vzdělávací účely. Podporuje operační systémy GNU/Linux, FreeBSD, Mac OS X. Je poskytován pod licencí GNU/GPLv2. (What is ns-3, © 2011-2012)
- **CORE** – Nástroj zajišťuje emulaci sítí na jedné nebo více stanicích. Emulované sítě je pak možné připojit do reálné sítě. Obsahuje také GUI pro návrh topologií sítí. Vyžaduje operační systémy Linux nebo FreeBSD. Více informací lze nalézt v (Ahrenholz, 2010).

Imalse je navržen tak, aby oddělil mechanismus botnetu a sítě. Jeho strukturu ilustruje Obrázek 5.2. Je založen na dvou konceptech:

- **Uzel** – Označení pro abstrakci reálné stanice.

[14] Celosvětový program společnosti Google, v němž uděluje stipendia studentům, kteří se přihlásí a úspěšně dokončí svůj projekt z oblasti vývoje softwaru během letních měsíců.

- **CMD** – Určuje sadu příkazů pro uzel. Definuje tak události, které může uzel generovat. CMD se dále dělí na 3 typy:
  - **Botmaster CMD** – Botmaster představuje speciální typ klienta s vyšším oprávněním. Zasílá příkaz k útoku na server.
  - **Server CMD** – Definuje server botnetu a jeho chování. Server vyčkává na příkaz k útoku od botmastru, následně odesílá příslušné příkazy klientům.
  - **Klient CMD** – Klient čeká na příkazy od serveru, na jejichž základě provádí příslušné akce na infikované stanici. Akce je definována v API uzlu.



Obrázek 5.2 – Struktura Imalse

Imalse je nový projekt, který však plánuje autor dále rozšířit. V současné době podporuje dva scénáře útoku:

1. **ddos_ping_flooding** – V tomto scénáři zašle botmaster příkaz serveru botnetu k zahájení útoku typu DDoS, konkrétně Ping flooding, na určený server.
2. **file_exfiltration** – Botmaster u tohoto scénáře zašle serveru botnetu požadavek, díky němuž server rozešle příkaz všem robotům na infikovaných stanicích k vyhledání souboru s konkrétním vzorem, například heslem. Jakmile bot nalezne požadovaný soubor, nahraje jej na určený server FTP.

Dále lze v Imalse využít modul experiment. Avšak jeho použití je omezeno pouze na simulační režim. Pomocí experimentu se vytváří topologie a konfigurace sítě a specifikace chování infikovaných stanic. Framework poskytuje tři typy experimentů:

1. **StaticRouteExperiment** – V tomto experimentu je manuálně přednastavena topologie ve tvaru diamantu a směrovací tabulky pro všechny uzly.
2. **TopoExperiment** – Pro daný experiment lze načíst topologii vygenerovanou například pomocí generátoru topologie např. Inet[15]. Adresy IP budou jednotlivým uzlům automaticky přiřazeny. Dále je možné nastavit síťovou adresu, adresu serveru, masku IP, server ID, botmaster ID, client ID, rychlost přenosu, zpoždění.

[15] Nástroj lze stáhnout z: http://topology.eecs.umich.edu/inet/

3. **ManualTopoExperiment** – Tento experiment je téměř shodný s TopoExperiment. Umožňuje však navíc nastavit rychlost přenosu a zpoždění pro každou linku zvlášť a také adresu IP pro každý uzel. Potřebný skript k nastavení topologie a chování uzlů je možné vygenerovat pomocí editoru GUI.

Pro jednodušší vytvoření topologie, definici chování a vizualizaci simulace lze využít dva grafické nástroje:

- **CORE Network Topology Editor** – S tímto editorem lze pohodlně navrhnout topologii sítě, přiřadit jednotlivým uzlům adresy IP, určit roli pro každý uzel, zpoždění, rychlost přenosu atd.
- **PyViz** – Vizualizační nástroj jenž je součástí simulátoru NS3. Umožňuje zobrazit topologii sítě, spuštění simulace a sledování probíhajícího provozu v navržené síti v reálném simulačním čase.

(Wang, 2012)

6 Praktická část

V této kapitole budou probrány praktické realizace vybraných síťových útoků spolu s popsáním jejich principů, postupů a možné obrany.

6.1 Instalace ns-3 a Imalse

Instalace byla provedena na operačním systému Ubuntu 13.10. Wang (2012) v dokumentaci doporučuje stáhnout balík simulátoru ns-3 (verze 3.14.1) s již implementovaným modulem pro Imalse. V terminálu operačního systému Ubuntu se stažení balíku provedlo pomocí tohoto příkazu:

```
wget https://bitbucket.org/hbhzwj/imalse/downloads/ns-allinone-3.14.1-with-imalse.tar.gz
```

Následně bylo třeba balík extrahovat a poté se přesunout do vzniklého adresáře:

```
tar -xzvf ns-allinone-3.14.1-with-imalse.tar.gz
cd ns-allinone-3.14.1-with-imalse
```

Simulátor ns-3 je napsán v jazyce C++, bylo tedy třeba mít nainstalovaný kompilátor pro tento jazyk např. *gcc*. Kompilace se standardně provádí s těmito parametry:

```
['-Wall'], ['-Werror'], ['-Wextra']
```

Při takto spuštěné kompilaci došlo k následující chybě:

```
../src/network/utils/ipv6-address.cc: In function ‘uint32_t
ns3::lookuphash(unsigned char*, uint32_t, uint32_t)’:
../src/network/utils/ipv6-address.cc:64:26: error: typedef ‘ub1’ locally
defined but not used [-Werror=unused-local-typedefs]
   typedef unsigned  char ub1;   /* unsigned 1-byte quantities */
                          ^
cc1plus: all warnings being treated as errors
Waf: Leaving directory `/run/media/danimoth/e6fe8c8a-2084-4bf3-9d12-
1750ff26b636/Work/ns-3-dev/build'
Build failed
 -> task in 'ns3-network' failed (exit status 1):
        {task 21381520: cxx ipv6-address.cc -> ipv6-address.cc.1.o}
['/usr/bin/g++', '-Wall', '-Werror', '-fPIC', '-pthread', '-I.', '-I..', '-
DNS3_ASSERT_ENABLE', '-DNS3_LOG_ENABLE', '-DHAVE_SYS_IOCTL_H=1', '-
DHAVE_IF_NETS_H=1', '-DHAVE_PACKET_H=1', '-DHAVE_SQLITE3=1', '-
DHAVE IF TUN H=1', '../src/network/utils/ipv6-address.cc', '-c', '-o',
'src/network/utils/ipv6-address.cc.1.o']
./waf -j8  346,70s user 21,06s system 722% cpu 50,889 total
```

Proto bylo nezbytné před samotnou kompilací editovat soubor *ipv6-address.cc*, který se nachází v adresáři *ns-allinone-3.14.1-with-imalse/ns-3.14.1/src/network/utils*. V tomto souboru bylo potřeba smazat řádek 63, kde se vyskytovala následující deklarace proměnné: *typedef unsigned char ub1;* Ta však není dále v kódu použita, a tak způsobuje při kompilaci varování, které se však převedlo dle parametrů na chybu.

Po úspěšné kompilaci, terminál zobrazil následující výstup:

```
'build' finished successfully (4m29.626s)
```

```
Modules built:
antenna                   aodv                      applications
bridge                    buildings                 config-store
core                      csma                      csma-layout
dsdv                      dsr                       emu
energy                    flow-monitor              imalse
internet                  lte                       mesh
mobility                  mpi                       netanim (no Python)
network                   nix-vector-routing        olsr
point-to-point            point-to-point-layout     propagation
spectrum                  stats                     tap-bridge
test (no Python)          tools                     topology-read
uan                       virtual-net-device        visualizer
wifi                      wimax

Modules not built:
click                     openflow

Leaving directory `./ns-3.14.1'
```

Další nezbytný krok představovalo editování souboru ns3.py, nacházejícího se v (*ns-allinone-3.14.1-with-imalse/ns-3.14.1/build/bindings/python/*) a zde zakomentování řádku 14:

```
# from ns.dsr import *
```

Tento řádek zajišťuje importování modulu ns.dsr, který však v ns-3.14.1 způsobuje bug.

Nakonec bylo třeba stáhnout z repozitáře samotný framework Imalse ze serveru https://bitbucket.org. Bylo však třeba mít nainstalován verzovací nástroj Mercurial. Stažení adresáře z repozitáře se provede příkazem:

```
hg clone https://bitbucket.org/hbhzwj/imalse
```

Ve vzniklém adresáři se nachází soubor *settings.py*. Tento soubor byl editován a upravena absolutní cesta v proměnné *ROOT* tak, aby odkazovala na tento aktuální adresář. Poté se ještě upravila absolutní cesta v proměnné *NS3_PATH*, kde se zadala cesta k adresáři simulátoru ns-3.

6.2 Instalace CORE

Prvním krokem při instalaci nástroje CORE bylo stažení balíčků *core-daemon_4.6-0ubuntu1_precise_amd64.deb* a *core-gui_4.6-0ubuntu1_precise_all.deb* z adresy: http://downloads.pf.itd.nrl.navy.mil/core/packages/4.6/.

Další nezbytnou část zahrnuje stažení balíčku s nástrojem Quagga, jenž slouží pro síťové směrování a obsahuje implementace směrovacích protokolů RIP, OSPF, BGP v několika verzích. Pro instalaci nástroje z terminálu slouží následující příkaz:

```
sudo apt-get install quagga
```

Nyní bylo třeba přesunout se do příslušného adresáře, kam se stáhly balíčky. Instalace balíčků se nejsnáze provádí pomocí grafického nástroje *Centrum softwaru pro Ubuntu*, jenž je standardně součástí systému. Nástroj automaticky zjistí závislosti k příslušnému balíčku a

případně je doinstaluje. V dalším kroku tedy stačilo zadat do terminálu následující příkazy a poté pomocí GUI oba balíčky nainstalovat:

```
software-center core-daemon_4.6-0ubuntu1_precise_amd64.deb
software-center core-gui_4.6-0ubuntu1_precise_all.deb
```

Grafické rozhraní aplikace CORE se spustí zadáním příkazu:

```
core-gui
```

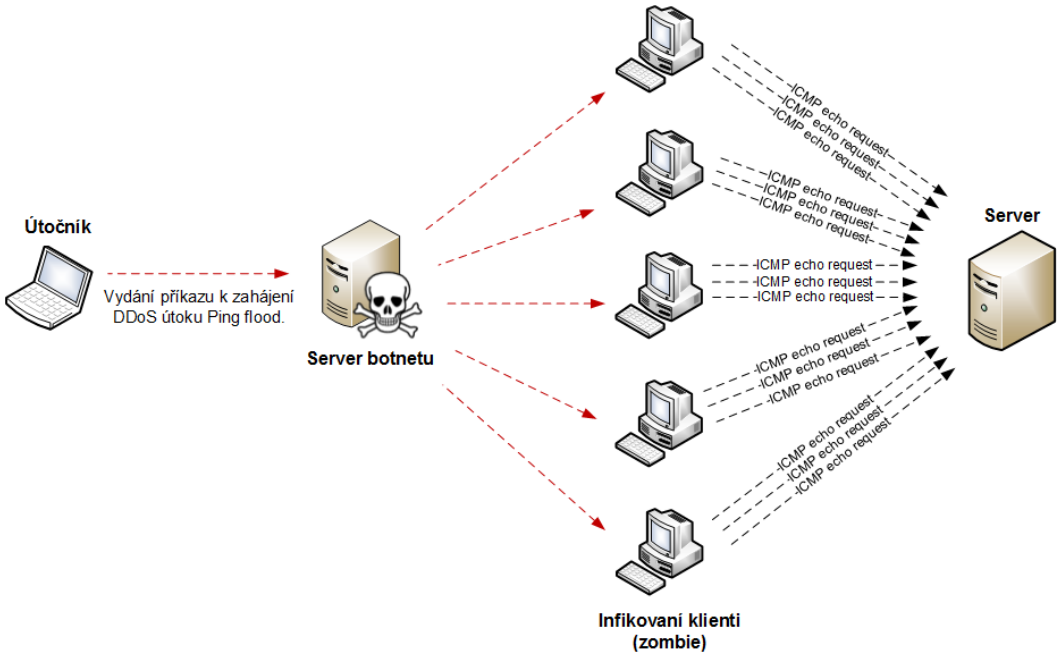
6.3 Simulace vybraných síťových útoků

V následujících pododdílech jsou postupně představeny vybrané síťové útoky z pohledu principu útoku, jeho realizace a způsobu obrany. Pro simulaci byly vybrány tyto síťové útoky: DDoS ping flooding, File exfiltration, SYN flood, SSL spoofing a E-mail phishing.

6.3.1 DDoS ping flooding

Princip útoku

Cílem útoku je pomocí botnetu (vysvětlen v kapitole 1.2.6) přetížit, případně úplně odstavit server, a tím také odepřít služby, které tento server poskytuje uživatelům. Útočník odešle na server botnetu příkaz k zahájení útoku. Server botnetu vydá jednotlivým infikovaným stanicím botnetu (tzv. zombie) pokyn k neustálému zasílání *ICMP echo requests* serveru (schéma útoku znázorňuje Obrázek 6.1). Jestliže je intenzita těchto požadavků dostatečně vysoká, server se přetíží a nebude schopen odpovídat na požadavky včas, případně může také dojít k jeho úplnému zhroucení.

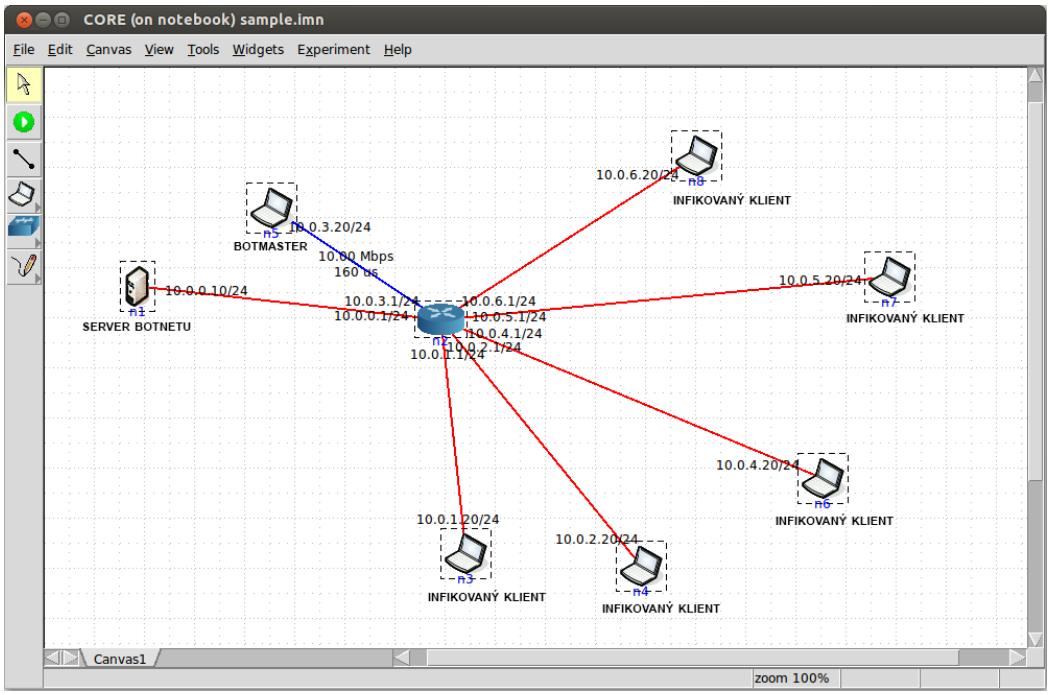


Obrázek 6.1 – Schéma útoku DDoS ping flooding

Realizace simulace

Simulace byla provedena za použití nástroje Imalse. Předem je třeba zmínit, že v této scénáři nástroj neprovádí samotnou simulaci útoku, ale simulaci komunikace mezi uzly botnetu při útoku.

Nejprve bylo potřeba vytvořit topologii sítě botnetu. V terminálu, přechodem do adresáře Imalse a následným zadáním příkazu *startgui.sh*, byl spuštěn nástroj CORE, modifikovaný přesně pro potřeby Imalse. Zde byla vytvořena topologie – Obrázek 6.2. U jednotlivých uzlů se nastavily příslušné role sítě botnet. Serveru navíc *traceflag* na hodnotu *yes*, pro zaznamenání paketů do souboru, generovaného při simulaci.



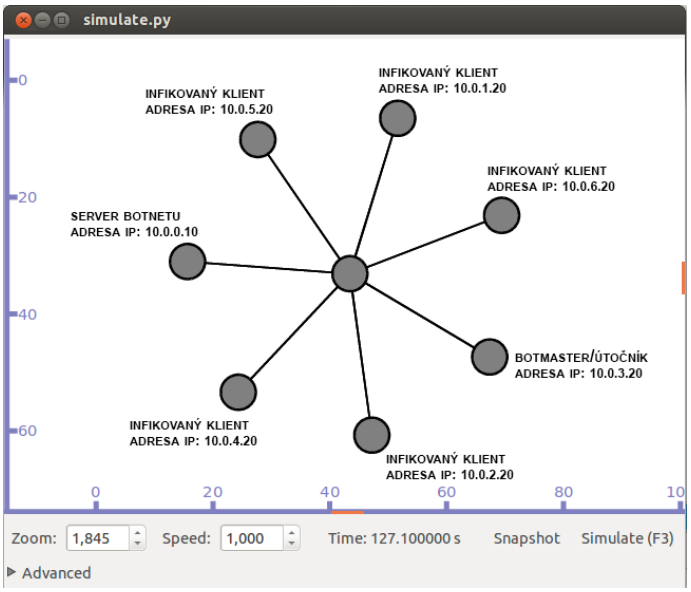
Obrázek 6.2 – Topologie sítě botnetu pro simulaci DDoS ping flooding

Pro vygenerování konfiguračních souborů navržené topologie bylo třeba v nástroji CORE zvolit záložku *File → Export Imalse config script*.

Dále byl editován skript *loadguiconfig.sh* v adresáři Imalse s těmito parametry:

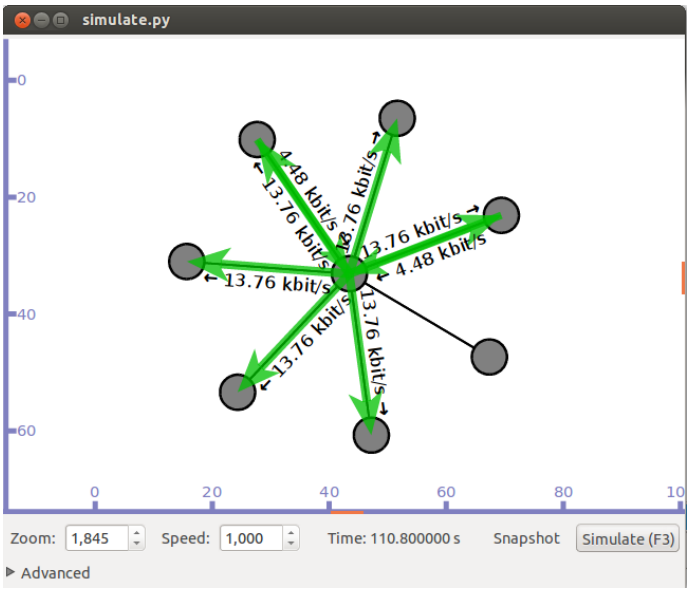
```
./sim -e ManualTopoExperiment --mode sim -s ddos_ping_flooding -f
\gui/imalse_config/topology.inet
\--net_settings=gui/imalse_config/net_settings.py \ -t 100
--SimulatorImpl=Visual
```

Posledním krokem před samotnou simulací představovalo spuštění nového terminálu a zadání příkazu: *loadguiconfig.sh*. Tím došlo ke spuštění nástroje PyViz s modelovou topologií, vizte Obrázek 6.3.



Obrázek 6.3 – Navržená topologie v nástroji PyViz

Poté již aktivováním tlačítka *Simulate (F3)* došlo k samotné simulaci provozu v navržené topologii, kde nástroj PyViz s využitím animace zobrazoval jednotlivé komunikace mezi uzly, jak je zachyceno (Obrázek 6.4).



Obrázek 6.4 –Simulace komunikace uzlů při útoku v nástroji PyViz

Po dokončení simulace došlo k vygenerování souborů se zachycenými pakety pro vybrané uzly (vizte příloha A). Tyto soubory je možné otevřít pomocí příslušných nástrojů např. Wireshark, který umožnuje zobrazit jednotlivé typy paketů odesílané a přijímané z daného uzlu. Lze tak pozorovat navázání spojení serveru botnet s infikovanými klienty tzv. zombie a udržování komunikace během simulace.

Obrana proti útoku

V tomto případě je nutné odlišovat obranu ze dvou aspektů. Prvním z nich je samotná obrana stanic klientů před infikováním boty. Doporučení pro uživatele je, aby udržovali svůj systém

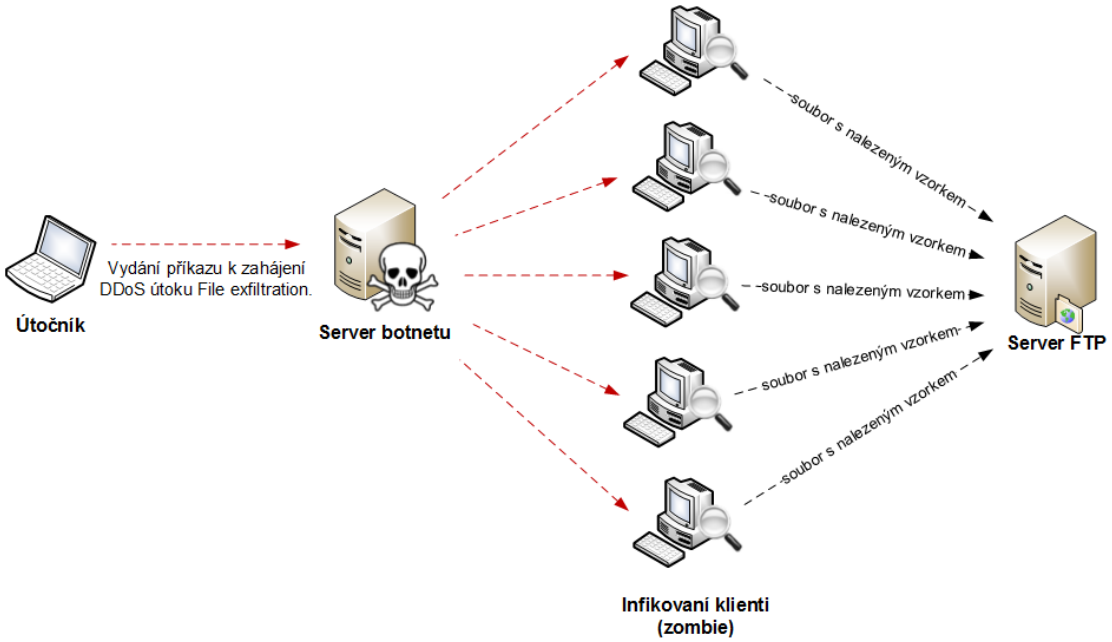
aktuální, používali kvalitní antivirový program, případně další nástroje bránící průniku malwaru a také nenavštěvovali potenciálně nebezpečné webové stránky.

Jak zabránit probíhajícímu útoku na server je poměrně náročné a nedá se zcela vymezit jedno účinné řešení. Většinou je zapotřebí spojit více obranných mechanismů. Nezbytné k odražení útoku je monitorování sítě a především zkušený správce, jenž dokáže podle situace správně reagovat na vývoj útoku. Účinné řešení představuje správně nastavený firewall na hraničním routeru. Efektivitu obrany také zvyšuje nasazení systému IPS.

6.3.2 File exfiltration

Princip útoku

Útočník zašle požadavek k útoku File exfiltration na server botnetu. Server botnetu poté zašle příkaz botům k prohledávání souborových systémů na jejich hostitelských infikovaných stanicích. Boti se snaží najít v souborech podle zadaného vzorku zajímavé údaje jako hesla, uživatelská jména atd. Pokud takovéto soubory naleznou, zasílají je na určený server FTP, odkud si je může útočník jednoduše stáhnout. Útok je schematicky znázorněn – Obrázek 6.5.



Obrázek 6.5 – Schéma útoku File exfiltration

Realizace simulace útoku

Opět byl pro tuto simulaci využit nástroj Imalse, avšak na rozdíl od předchozí simulace, se použil emulační režim, který umožňuje provést simulaci útoku reálně, bez simulátoru ns-3.

Simulace útoku vyžaduje použít server FTP. V tomto případě byl nainstalován server vsftpd přímo na operačním systému Ubuntu 13.10. Další část představovalo editování soubor *config.py* (cesta: *imalse/scenario/file_exfiltration/*), jak ukazuje následující výstup:

```
SRV_ADDR = '127.0.0.1'
'ftp':{
   'host':'localhost',
   'user':'radek',
   'password':'heslo',
   },
'file_filter':{
   'suffix':['.txt'],
   'pattern':'assword',
   'directory':'../'
   }
```

Atributy v bloku *'ftp'* představují přihlašovací údaje k serveru FTP, blok *'file_filter'* obsahuje parametry pro vyhledání shodujících se souborů v souborových systémech infikovaných klientů. Důležitým parametrem je především *'pattern'*, jenž vyhledává v příslušných souborech zadaný textový řetězec v tomto případě *'assword'*.

Pro potřeby simulace bylo třeba vytvořit pokusné soubory s příponou *txt*, do nichž byl zapsán řetězec *password* a umístěny na různá místa v uživatelském adresáři.

Poslední úkon před samotnou simulací útoku představovalo spuštění tří terminálů a do každého byl zadán jeden z následujících příkazů:

```
./emulate.py -s file_exfiltration -r server
./emulate.py -s file_exfiltration -r client
./emulate.py -s file_exfiltration -r botmaster
```

Přepínač *-r* určuje roli, která je simulována, vizte kapitola 5.5. Po zadání příkazu s přepínačem *-r botmaster* došlo k provedení útoku. Terminál s příkazem obsahující přepínač *-r server* zobrazil následující výstup:

```
addr_port,  ('127.0.0.1', 3333)

sock_num 1
broke_socks []
result,  {"host": "localhost", "password": "heslo", "user": "radek",
"event": "set_ftp_info"}
sock_num 1
broke_socks []
result,  {"directory": "../", "pattern": "assword", "suffix": [".txt"],
"event": "set_file_filter"}
sock_num 1
broke_socks []
result,  {"event": "search_and_upload"}
```

Nejdůležitějším výstupem v terminálu klienta představoval výsledek o nalezení souborů s hledaným vzorkem:

```
interesting_files
['/home/radek/imalse/scenario/file_exfiltration/test.txt',
'/home/radek/Plocha/dokument.txt', '/home/radek/Hudba/readme.txt',
'/home/radek/Dokumenty/soubor_test.txt',
'/home/radek/Dokumenty/imalse/imalse/scenario/file_exfiltration/test.txt']
```

Výstup mimo jiné obsahoval seznam všech prohledávaných souborů a informace o odeslání souborů s nalezeným vzorkem na server FTP. Celý konzolový výstup z klientského terminálu se nachází v příloze A.

Simulaci útoku by bylo možné také realizovat na více stanicích v síti (představující infikované klienty), za předpokladu, že by tyto stanice měly nainstalován nástroj Imalse.

Obrana proti útoku

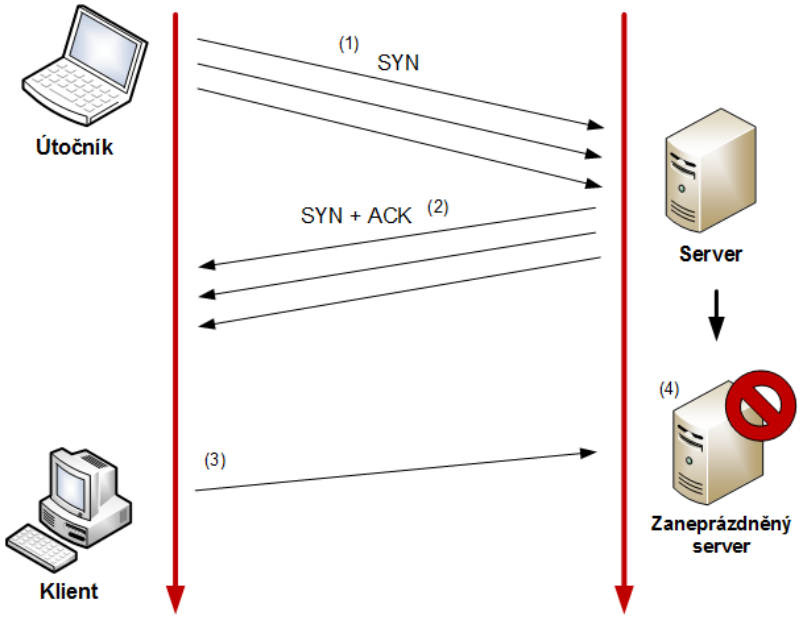
Stejně jako u předchozího útoku, neexistuje přesný návod, jak zabránit botům před infikováním stanic uživatelů. Obecně lze však opět doporučit oprostit se od návštěvy nebezpečných webů, pravidelně aktualizovat operační systém, mít nainstalován kvalitní antivirus, personální firewall a základní nástroje pro detekci škodlivého malwaru.

6.3.3 SYN flood (DoS)

SYN flood patří k nejčastěji realizovaným útokům typu DDoS, jak ilustruje Obrázek 1.6.

Princip útoku

Útok je schematicky znázorněn (Obrázek 6.6). Útočník odesílá na server neustále pakety s příznakem SYN (1). Server odesílá zpět útočníkovi pakety s příznakem SYN a ACK (2). Poté vyčkává na pakety s příznakem ACK od útočníka, ty však nedostane. V bufferu serveru tak zůstávají alokována polootevřená spojení, která se zde hromadí do té doby, dokud není paměť zcela vyčerpána. Jakmile dojde k úplnému zaplnění bufferu, server přestává zpracovávat další požadavky od ostatních klientů (3), případně dojde k jeho pádu. Služby serveru se tak stávají nedostupné (4).



Obrázek 6.6 – Schéma útoku SYN flood

Realizace simulace útoku

Útok byl realizován v operačním systému Kali Linux, kde je předinstalovaný nástroj hping3 pro generování paketů, analýzu TCP/IP protokolů, bezpečnostní testování a audit. Díky velkému množství přepínačů, jejichž kompletní seznam včetně popisů lze nalézt v manuálových stránkách příkazem *man hping3* v terminálu, se jedná o velice mocný nástroj. Pro spuštění útoku SYN flood je třeba zadat v terminálu příkaz:

```
sudo hping3 --flood -S -p 80 IP_oběti
```

Přepínač *--flood* slouží pro neustálé generování a odesílání paketů, *-S* přidává každému paketu příznak SYN, *-p 80* znamená, že tyto pakety zasílány na cílový port 80, *IP_oběti* definuje adresu IP zařízení, na které má být útok proveden.

Při simulaci byl útok směrován na zařízení TP-LINK Wireless Lite N Router WR740N. Router se podařilo dostatečně zatížit tak, že jeho časové odezvy se několikanásobně zvýšily a docházelo k odmítání požadavků HTTP. Konzolový výstup nástroje ping z terminálu je možné shlédnout v příloze A.

Z důvodu toho, že útoky typu DoS mají na výkonná zařízení jako je server minimální dopady, bylo využito zmíněné zařízení. Na servery a ostatní výkonná zařízení je zapotřebí provést útok typu DDoS.

Obrana proti útoku

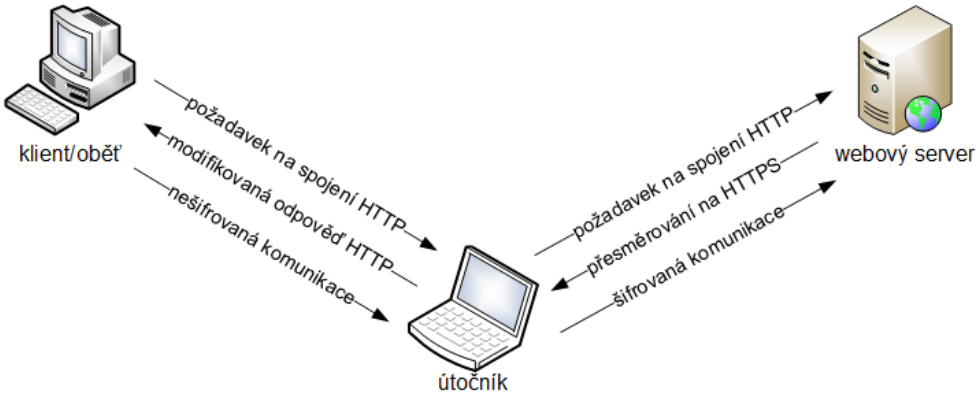
Obrana je v tomto případě dosti problematická. Krause (1999) uvádí, že možností je více, avšak žádná nenabízí naprostou eliminaci útoku. Zvětšení kapacitního prostoru pro poloviční handshake spojení či snížení „timeout“ těchto spojení lze navýšením intenzity zaplavování paketů překonat. Lépe tomuto útoku dokáží čelit IDS nebo inteligentní filtrování paketů.

Poslední účinnou metodou, jak se bránit útoku, je využití SYN cookies. Server vypočítá z údajů v paketu hodnotu hash, poté co odešle paket s příznakem SYN a paket s příznakem ACK ke klientovi, polootevřené spojení zahazuje. Pokud obdrží od klienta paket s příznakem ACK se sekvenčním číslem, odpovídající hodnotě hash, spojení otevře. Nevýhodou je vyšší výpočetní náročnost u této metody.

6.3.4 SSL spoofing (MITM)

Princip útoku

V roce 2009 vytvořil a představil počítačový bezpečnostní výzkumník, Moxie Marlinspike, skript (SSLstrip), napsaný v jazyce Python, jenž umožňuje při útoku typu MITM (vizte 1.2.3) podsunout oběti nešifrované spojení protokolem HTTP, namísto původně požadovaného šifrovaného protokolem HTTPS. Schématické znázornění útoku – Obrázek 6.7. (Peifer, 2010)



Obrázek 6.7 – SSL spoofing (podsunutí nešifrovaného spojení)

Realizace simulace útoku

K útoku, jak ukazuje Obrázek 6.7, je využít útok typu MITM. Je zapotřebí spustit terminál a zadat příkaz:

```
ettercap -T -q -i eth0 -M arp:remote /IP_oběti/ /IP_výchozí_brány/
```

Příkaz spustí nástroj Ettercap pro zachytávání paketů. Parametr *-T* zajistí spuštění v textovém (konzolovém) režimu, *-q* vypisuje pouze nejdůležitější informace, přepínač *-i* určuje rozhraní, ze kterého mají být pakety zachytávány. Ettercap obsahuje i modul pro MITM útoky, spuštění tohoto útoku provádí přepínač *-M*, v tomto případě je vybrán konkrétně útok ARP cache poisoning (*arp:remote*) následuje zadání adresy IP oběti a adresy IP výchozí brány.

V dalším terminálu, aby mohl být útok typu MITM realizován, je potřeba zjistit, zda je aktivní řetězec pro přeposílání paketů příkazem:

```
cat /proc/sys/net/ipv4/ip_forward
```

Pokud je *ip_forward* aktivní vypíše se číslo „1“ v opačném případě „0“. Pro povolení *ip_forward* slouží příkaz:

```
echo 1 >> /proc/sys/net/ipv4/ip_forward
```

Dalším nutným krokem pro úspěšnou realizaci útoku je zajištění přesměrování provozu HTTP z portu 80 na port, na kterém bude naslouchat skript sslstrip, např. 10000. Toto přesměrování zajistí nástroj iptables příkazem:

```
iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT   --
to-port 10000
```

Poslední částí z pohledu útočníka je otevření nového terminálu a zadání:

```
sslstrip -l 10000
```

Tím je zajištěno, že skript bude naslouchat na portu 10000 a přijímat tak komunikaci HTTP.

Nyní již stačí, aby oběť útoku navštívila server se zabezpečenou komunikací pomocí protokolu SSL/TLS a zadal své přihlašovací údaje. Útočník pak tyto údaje zachytí pomocí nástroje Ettercap. Výpis z terminálů realizované simulace útoku je k nalezení v příloze A.

Obrana proti útoku

Obranou před útokem SSL spoofing je především zvýšená pozornost při návštěvě webů, které komunikují šifrovaně, aby toto spojení bylo skutečně šifrované. To lze zjistit z adresního řádku prohlížeče, kde je nutné se přesvědčit, že je protokol HTTPS aktivní, tedy, že před adresou je označení tohoto protokolu (https://).

Reakcí na útoky SSL spoofing bylo také vytvoření hlavičky HSTS (HTTP Strict Transport Security), jak ve svém článku popisuje Cvrček (2012). Jedná se o obranný mechanismus, s jehož pomocí server, který umožňuje šifrované spojení, zasílá tyto hlavičky v odpovědi na požadavek HTTP od webového prohlížeče klienta. Dochází tak k vynucení spojní prostřednictvím protokolu HTTPS, a to i v případě, že uživatel zadal v adresním řádku prohlížeče spojení pomocí nešifrovaného protokolu HTTP (http://). Tím je tedy zabráněno v podvržení nešifrovaného spojení. Nutným předpokladem, pro fungování tohoto bezpečnostního mechanismu, je podpora hlaviček HSTS nejen na straně serveru, ale rovněž i na webovém prohlížeči. Například Mozilla Firefox podporuje hlavičku HSTS od verze 17 a Google Chrome od verze 4.

6.3.5 E-mail phishing

Princip útoku

Útočník rozesílá obětem podvržené e-maily, které se tváří jako autentické žádosti od bank, úřadů a dalších společností mající s oběťmi souvislost. Účelem těchto žádostí bývá přimění obětí k přístupu na falešné webové stránky, vizuálně totožné s autentickými stránkami, kde mají provést nejčastěji přihlášení do systému nebo vyplnění osobních údajů. Útočník pak tyto údaje zneužívá ve svůj prospěch.

Realizace simulace útoku

Simulace útoku byla provedena v operačním systému Kali Linux, za použití nástroje *setoolkit*.

V terminálu je třeba vybrat z menu položky:

```
Social-Engeneering Attacks → Website Attack Vectors → Tabnabbing Attack
Method → Site Cloner
```

Nástroj poté vyzve k zadání adresy IP, na kterou bude zaslán požadavek POST s uživatelským jménem a heslem. Následující výzva požaduje vyplnit adresu pro naklonování přihlašovací stránky. Pro tuto simulaci tak byla použita adresa https://gmail.com.

Takto naklonovaná stránka bude přístupná přes adresu IP, zadanou v předchozí výzvě. Je důležité zmínit, že pro simulaci byla použita privátní adresa, jelikož test probíhal na stejné síti. V praxi se pro reálný útok využívá veřejná adresa IP, útočník totiž zasílá podvržené e-

maily uživatelům, kteří se nacházejí v různých sítích. Po zadání všech parametrů musí zůstat tento terminál otevřený, budou sem přicházet požadavky POST s přístupovými údaji.

V dalším kroku, pro zaslání podvrženého e-mailu, je potřeba opět spustit nástroj *setoolkit* v terminálu. Zvolit tyto položky:

```
Social-Engeneering Attacks → Mass Mailer Attack
```

Podle potřeby vybrat: *E-mail Attack Single Email Address* (pro odeslání jednoho e-mailu) nebo *E-mail Attack Mass Mailer* (pro odeslání vice e-mailů).

Následuje výběr serveru SMTP. Je možné vybrat předpřipravený Gmail server SMTP, kde stačí zadat přihlašovací údaje k tomuto účtu nebo vlastní server SMTP. U vlastního serveru lze zadat libovolnou (falešnou) e-mailovou adresu. Nástroj poté již požádá o jméno odesílatele, předmět zprávy a text zprávy. Vhodným textem je třeba přimět příjemce ke kliknutí na odkaz (s adresou IP na útočníka) a zadání přístupových údajů. Po vyplnění údajů se tyto údaje zobrazí v druhém terminálu, což je známka úspěšné realizace útoku. Konzolový výstup simulace útoku je rovněž k nalezení v příloze A.

Obrana proti útoku

Příčinou rozesílání podvodných e-mailů jsou bezpečnostní nedokonalosti protokolu SMTP. Server nemá možnost zjistit při přijetí zprávy, zda odesílatel, uvedený v hlavičce této zprávy, je skutečně tím za koho se vydává. Obvykle se kontroluje pouze správná syntaxe adresy a existence domény.

Možností, jak zajistit autentičnost odesílatele je využít elektronický podpis, uvádí ve svém článku Valášek (2012). Nejlepší prevencí, jak předejít těmto útokům, je za žádných okolností nereagovat na podobné žádosti o zaslání uživatelských údajů. Skuteční administrátoři serverů nikdy nepožadují zasílat přístupové údaje. Také je tento útok dobře rozpoznatelný po kliknutí na odkaz v e-mailu. V adresním řádku se adresa nebude shodovat s adresou skutečného serveru.

7 Závěr

Cílem této práce bylo v teoretické části rozdělit typy útoků, popsat vrstvy modelu ISO/OSI a TCP/IP, na jejichž slabá místa jsou prováděny útoky. Další kapitola pak rozebírá nejdůležitější protokoly zajišťující bezpečnou síťovou komunikaci. Konec teoretické části se věnuje obecným možnostem zabezpečení síťové infrastruktury. Jelikož Internet představuje nejčastější místo, odkud jsou realizovány nelegální praktiky crackerů vůči síťovým infrastrukturám (například útoky File exfiltration, SYN flood, které byly simulovány), z toho důvodu problematika bezpečnosti zaujímá důležitou část při návrhu a správě komunikačních sítí.

Mnohem snadněji však lze velkou část útoků provádět přímo uvnitř sítí (příkladem jsou simulované útoky SSL spoofing a E-mail phishing). Proto by se neměl opomíjet ani takový scénář, jako útok vedený nespokojeným zaměstnancem na ostatní zaměstnance nebo nadřízené, ale i důležité systémy či servery nacházející se přímo uvnitř podnikové sítě. Je tak nezbytné nastavit vhodnou bezpečnostní politiku uvnitř sítě. Možností, jak otestovat odolnost a případná slabá místa umožňují penetrační testy, postupující podle metodik, například OWASP.

Praktická část popisuje instalaci nástroje Imalse a simulace útoků, vizte níže, za použití nástrojů Imalse, hping3, ettercap a setoolkit. Pomocí nástroje Imlase, umožňujícího vytvoření sítě botnet a simulaci implementovaných scénářů útoků, byla provedena simulace průběhu komunikace při útoku DDoS ping flooding, kdy došlo nejprve k vytvoření topologie botnetu ve vizualizačním nástroji CORE a následně byla tato konfigurace exportována do Imalse. Průběh simulace zobrazovala animace probíhající komunikace v botnetu. Výstupem pak byl soubor se zachycenými pakety při komunikaci mezi vybranými uzly, to umožňuje detailní pohled na vzájemnou interakci stanic v botnetu.

Nástroj Imalse byl použit také pro útok File exfiltration, jenž v souborovém systému infikované stanice sítě botnet vyhledával, podle zadaného vyhledávacího vzorku („assword“), soubory shodující se s tímto vzorkem. Nalezené soubory se následně zasílaly na předem definovaný server FTP. Infikovanou stanicí se může velmi snadno stát každá stanice, jejíž operační systém není dostatečně chráněn softwarem proti malwaru, zabraňující infikování botem.

Z distribuce Kali Linux byl nasazen nástroj hping3 k provedení simulace DoS útoku SYN flood na bezdrátový router TP-LINK Wireless Lite N Router WR740N, který se tímto útokem podařilo zatížit a došlo k odmítnutí požadavků, jenž byly poté na router zasílány. Dalším útokem, k němuž se využilo nástroje ettercap a skript SSLstrip z distribuce Kali Linux představoval útok SSL spoofing, s nímž bylo dokázáno, že pokud prohlížeč, případně server nepodporují bezpečnostní hlavičky HSTS, lze velmi snadno podvrhnout šifrované spojení za nešifrované na základě útoku typu MITM a útočník tak může takovou komunikaci bez problémů odposlouchávat.

Posledním simulovaným útokem byl tzv. sociální útok E-mail phishing. S pomocí nástroje *setoolkit*, obsaženého v Kali Linux, byl zaslán falešný e-mail jménem bezpečnostního administrátora Google s upozorněním na možné bezpečnostní ohrožení a výzvou na zadání uživatelského jména a hesla do podvržené přihlašovací stránky společnosti Google, jenž vytvářela vizuální kopii skutečné přihlašovací stránky Google. Po zadání těchto údajů byly tyto údaje odchyceny nástrojem ettercap na počítači, simulující útočníkovu stanici. Útok využívá nedokonalosti protokolu SMTP, který nedisponuje možností ověření odesílatele e-mailu.

Jak je z praktické části patrné, lze provést útoky za použití vhodných nástrojů. Útočník však musí mít určité teoretické znalosti o principech fungování síťové komunikace a rovněž detailní znalost systému, včetně jeho chyb, do kterého má v úmyslu proniknout. Nástroje použité pro simulování útoků jsou open-source, tím i volně dostupné pro veřejnost. Ačkoli jsou nástroje určeny k penetračnímu testování, dají se rovněž zneužít k provedení útoků.

Potenciál Imalse se ukrývá především v emulačním režimu. Tento nástroj představuje vhodnou podporu pro výzkum nových obranných mechanismů proti útokům, které je možné provádět pomocí botnetu. Také by se nástroj dal využít jako výukový materiál v síťových laboratořích pro ukázky principů fungování sítí botnet. Nevýhodou tohoto nástroje však je jeho náročnější ovládání a malé množství scénářů simulací útoků. Pokud by však tento projekt dále rozvíjela širší komunita vývojářů, jistě by nástroj nabyl větších možností.

8 Literatura

AHRENHOLZ, J., 2010. Comparison of CORE Network Emulation Platforms. In: *Conference record / IEEE Military Communications Conference*. s. 166–171. ISBN 978-1-4244-8178-1. ISSN 2155-7578. Dostupné z: http://ieeexplore.ieee.org/iel5/5673920/5679517/05680218.pdf

CALETKA, Ondřej, 2014. Heartbleed bug: vážná zranitelnost v OpenSSL. *Root.cz* [online]. 11. 4. 2014 [cit. 2014-04-14]. ISSN 1212-8309. Dostupné z: http://www.root.cz/clanky/heartbleed-bug-vazna-zranitelnost-v-openssl/

CANONICAL LTD., © 2014. About Ubuntu. *Ubuntu* [online]. [cit. 2014-03-16]. Dostupné z: http://www.ubuntu.com/about

CISCO SYSTEMS INC., 2009. What Is the Difference: Viruses, Worms, Trojans, and Bots?. *Cisco Systems Inc.* [online]. [cit. 2014-04-12]. Dostupné z: http://www.cisco.com/web/about/security/intelligence/virus-worm-diffs.html

CVRČEK, Pavel, 2012. S Firefoxem 17 přistupujete ke konkrétním webům bezpečněji. *Mozilla.cz* [online]. 25. 11. 2012 [2014-04-14]. Dostupné z: http://www.mozilla.cz/zpravicky/s-firefoxem-17-pristupujete-ke-konkretnim-webum-bezpecneji/

ČERNÝ, Michal, 2010. *Systémy detekce a prevence průniku*. Brno [online]. [cit. 2014-03-09]. Diplomová práce. 85 s. Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Vedoucí práce Ing. Radim Pust. Dostupné z: https://dspace.vutbr.cz/bitstream/handle/11012/5335/DP.pdf

DOČEKAL, Michal, 2010. Správa linuxového serveru: Pár slov o útocích na server. *Linux EXPRES* [online]. 23. 9. 2010, [cit. 2014-02-16]. ISSN 1801-3996. Dostupné z: http://www.linuxexpres.cz/praxe/sprava-linuxoveho-serveru-par-slov-o-utocich-na-server

HOLUB, Petr, 2002. Jemný úvod do (anti)virové problematiky. *Zpravodaj ÚVT MU* [online]. 2002, roč. XII, č. 4, s. 9-14 [cit. 2014-02-18]. ISSN 1212-0901. Dostupné z: http://www.ics.muni.cz/bulletin/articles/245.html

KRAUSE, Michal, 1999. Noční můra jménem SYN flooding. *Root.cz* [online]. 29. 7. 1999 [cit. 2014-04-12]. ISSN 1212-8309. Dostupné z: http://www.root.cz/clanky/nocni-mura-jmenem-syn-flooding/

LAMMLE, Todd, 2010. *CCNA: výukový průvodce přípravou na zkoušku 640-802*. Vyd. 1. Brno: Computer Press, 928 s. ISBN 978-802-5123-591.

MCDOWELL, Mindi, 2009. Avoiding Social Engineering and Phishing Attacks. *U. S. Department of Homeland Security: US-CERT* [online]. 22 October 2009 [cit. 2014-02-17]. Dostupné z: http://www.us-cert.gov/ncas/tips/ST04-014

NELSON, Chad, 2011. Cyber Warfare: The Newest Battlefield. *Washington University in St. Louis* [online]. 27 November 2011 [cit. 2014-02-17]. Dostupné z: http://www1.cse.wustl.edu/~jain/cse571-11/ftp/cyberwar/index.html

NORTHCUTT, Stephen et al., 2005. *Bezpečnost sítí: velká kniha*. Vyd. 1. Brno: Computer Press, 589 s. ISBN 80-251-0697-7.

OFFENSIVE SECURITY LTD., 2013. What is Kali Linux?, 2013. *Kali Linux Official Documentation* [online]. 25 February 2013 [cit. 2014-03-16]. Dostupné z: http://docs.kali.org/introduction/what-is-kali-linux

ODVÁRKA, Petr, 2001. UDP protokol. *Svět sítí* [online]. 4. 1. 2001 [cit. 2014-02-16]. Dostupné z: http://www.svetsiti.cz/clanek.asp?cid=UDP-protokol-412001

ODVÁRKA, Petr, 2002a. SSL protokol (3) - SSL Handshake Protocol. *Svět sítí* [online]. 9. 5. 2002 [cit. 2014-03-05]. Dostupné z: http://www.svetsiti.cz/clanek.asp?cid=SSL-protokol-3-SSL-Handshake-Protocol-952002

ODVÁRKA, Petr, 2002b. SSL protokol (4) - Change Cipher Spec Protocol a Alert Protocol. *Svět sítí* [online]. 16. 5. 2002 [cit. 2014-03-05]. Dostupné z: http://www.svetsiti.cz/clanek.asp?cid=SSL-protokol-4-Change-Cipher-Spec-Protocol-a-Alert-Protocol-1652002

ODVÁRKA, Petr, 2002c. SSL protokol (5) - SSL Record Protocol. *Svět sítí* [online]. 21. 5. 2002 [cit. 2014-03-05]. Dostupné z: http://www.svetsiti.cz/clanek.asp?cid=SSL-protokol-5-SSL-Record-Protocol-2152002

ORACLE CORPORATION, © 2004-2014. Chapter 1. First steps. *Documentation – Oracle VM VirtualBox* [online]. [cit. 2014-03-16]. Dostupné z: https://www.virtualbox.org/manual/ch01.html

OSTERLOH, Heather, 2003. *TCP/IP: kompletní průvodce : použitelný pro veškeré operační systémy*. Praha: SoftPress, 512 s. ISBN 80-864-9734-8.

PASSERI, Paolo, 2013. August 2013 Cyber Attacks Statistics. *Hackmageddon.com* [online]. 7 September 2013 [cit. 2014-02-18]. Dostupné z: http://hackmageddon.com/2013/09/07/august-2013-cyber-attacks-statistics/

PEIFER, Duane, 2010. SSL Spoofing: Man-In-The-Middle attack on SSL. *OWASP* [online]. [cit. 2014-04-15]. Dostupné z: https://www.owasp.org/images/7/7a/SSL_Spoofing.pdf

PETERKA, Jiří, 1992a. Síťový model TCP/IP. *EArchiv.cz* [online]. [cit. 2014-02-16]. Dostupné z: http://www.earchiv.cz/a92/a231c110.php3

PETERKA, Jiří, 1992b. Transportní protokoly TCP/IP - I. *EArchiv.cz* [online].[cit. 2014-02-16]. Dostupné z: http://www.earchiv.cz/a92/a250c110.php3

PETERKA, Jiří, 2001. Principy firewallů: Paketové filtry vs aplikační brány. *EArchiv.cz* [online]. [cit. 2014-02-16]. Dostupné z: http://www.earchiv.cz/b01/b0100024.php3

PROLEXIC TECHNOLOGIES INC., 2014. Prolexic Quarterly Global DDoS Attack Report Q4 2013. *Prolexic Technologies* [online]. [cit. 2014-04-17]. Dostupné z: http://www.prolexic.com/kcresources/attack-report/prolexic-quarterly-global-ddos-attack-report-q413/Prolexic-Q42013-Global-Attack-Report-A4-011314-2.pdf

PŘIBYL, Tomáš, 2006. Zákeřný útok jménem DoS. *IT Systems* [online]. 11/2006 [cit. 2014-02-17]. ISSN 1802-615X. Dostupné z: http://www.systemonline.cz/it-security/zakerny-utok-jmenem-dos.htm

PUŽMANOVÁ, Rita, 2006. *Moderní komunikační sítě od A do Z*. Vyd. 2. Brno: Computer Press, 430 s. ISBN 80-251-1278-0.

SOSINSKY, Barrie A., 2010. *Mistrovství – počítačové sítě*. Vyd. 1. Brno: Computer Press, 840 s. ISBN 978-80-251-3363-7.

VALÁŠEK, Michal A., 2012. Hejtmanem za pět minut: jak snadné je podvrhnout odesílatele e-mailové zprávy? IHNed [online]. 5. 10. 2012 [cit. 2014-02-17]. ISSN 1213–7693 Dostupné z: http://tech.ihned.cz/c1-57770760-hejtmanem-za-pet-minut-jak-snadne-je-podvrhnout-odesilatele-e-mailove-zpravy

WANG, Jing Conan, 2012. Imalse 0.0 alpha documentation. *Boston University* [online]. [cit. 2014-03-16]. Dostupné z: http://people.bu.edu/wangjing/open-source/imalse/html/index.html

What is ns-3, © 2011-2012. *ns-3* [online]. [cit. 2014-03-17]. Dostupné z: http://www.nsnam.org/overview/what-is-ns-3/

9 Přílohy

Příloha A – CD, obsahující:

- výstupy z jednotlivých simulací,
- elektronická verze bakalářské práce v PDF.