VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2017 Bc. Martin Šindler

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

LABORATORNÍ ÚLOHA ÚTOKŮ NA PROTOKOL HTTPS
LABORATORY EXERCISE OF ATTACKS ON HTTPS PROTOCOL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Martin Šindler**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Karel Burda, CSc.**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Martin Šindler ***ID:*** 154667
Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Laboratorní úloha útoků na protokol HTTPS

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte a popište problematiku protokolu HTTPS a útoků na tento protokol. Na tomto základě navrhněte laboratorní úlohy pro vybrané útoky uskutečnitelné na jediném počítači ve virtuální síti VMware. Pro úlohy připravte software a zpracujte pro ně výukovou dokumentaci jak pro studenty, tak i vyučujícího. Při tvorbě dokumentace pro studenty respektujte didaktické zásady.

DOPORUČENÁ LITERATURA:

[1] Rescorla E.: HTTP Over TLS. RFC 2818. IETF, Fremont 2000.

[2] Khare R., Lawrence S.: Upgrading to TLS Within HTTP/1.1. RFC 2616. IETF, Fremont 2000.

Termín zadání: 1.2.2017 ***Termín odevzdání:*** 24.5.2017

Vedoucí práce: doc. Ing. Karel Burda, CSc.
Konzultant:

doc. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Tato diplomová práce popisuje princip vybraných útoků na HTTPS a následně realizuje jednotlivé útoky prostřednictvím laboratorních úloh založených na platformě VMware. Vypracování obsahuje oddělené části zabývající se specifickým útokem, hodnocením rizika zranitelnosti a popisu způsobu provedení. Vypracování obsahuje také postupy k laboratorním úlohám a dokumentaci pro vyučující.

KLÍČOVÁ SLOVA

tls, ssl, spoof, renegotiation, poodle, soukromý klíč, heartbleed, útok, cbc, riziko, laboratorní, úlohy

ABSTRACT

This thesis describes principles of specific HTTPS attacks and provides realization of attacks using lab environment based on VMware. Elaboration is divided to specific dedicated attacks, each part evaluates risk of vulnerability and describes the way of realization in guides for students and lector´s manuals.

KEYWORDS

tls, ssl, spoof, renegotiation, poodle, private key, heartbleed, attack, cbc, risk, labs

ŠINDLER, Martin *Laboratorní úloha útoků na protokol HTTPS*: Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2017. 98 s. Vedoucí práce byl doc. Ing. Karel Burda, CSc.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Laboratorní úloha útoků na protokol HTTPS“ jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu doc. Ing. Karel Burda, CSc za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .

podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Během devadesátých let se způsob využívání internetu měnil. Původně nekomerční prostředí přestávalo být tolik dominantní a zesiloval vliv komerčních služeb. Toto rozšiřování tedy proto doprovázelo zvyšování požadavků na bezpečnost, která byla na velmi špatné úrovni. Hlavním důvodem byla změna filozofie využívání osobních počítačů, které začaly být během poslední dekády dvacátého století prostředkem pro komunikaci na internetu. Dynamická změna prostředí měla za následek velké množství bezpečnostních problémů, které nemohly být ponechány bez povšimnutí společnostmi zaměřenými na bezpečnost.

Na začátku roku 1995 byla vydána společností Netscape první verze protokolu SSL, tedy Secure Socket Layer[1], která představovala jeden z prvních zásadních kroků zabezpečení komunikace nejen pro vládu, ale především pro komerční služby. Úspěch protokolu SSL vycházel zejména ze dvou důležitých faktů.

První z nich byl vývoj SSL, který byl v prvních letech vývoje velice dobře chráněn, zejména kvůli intelektuálnímu vlastnictví. Společnost Netscape usilovně bránila jakýkoliv únik informací, především z obavy, že by konkurence využila poptávku, která zatím nebyla přesně specifikovaná. Poté, co byl v roce 1997 registrován patent, společnost téměř okamžitě publikovala specifikaci volně k použití.

Druhým zásadním důvodem rychlého rozšíření byla spolupráce s organizací IETF, která využila SSL 3.0 jako vhodný standard pro webové zabezpečení a uvedla protokol pod jménem TLS, tedy Transport Layer Security [15].

Nejobtížnějším krokem však bylo přesvědčit velké společnosti jako např. Microsoft k podpoře SSL a TLS protokolů. Implementace do komerčních operačních systémů představovala zásadní krok nejen k rozšíření samotného protokolu, ale měla také velký vliv na zabezpečení běžných uživatelů.

Rozšíření SSL/TLS mělo za následek podrobnou analýzu potenciálních zranitelností ze strany bezpečnostních expertů, kteří byli motivováni různými důvody. Ať už byly důvody jakékoliv, postupné řešení bezpečnostních problémů dopomohlo k lepším implementacím těchto protokolů.

V současnosti je vnímáno SSL/TLS velice kontroverzně. Široká veřejnost rozumí významu tohoto slovního spojení jen velmi málo a spíše ho považuje za nepřekonatelnou záruku bezpečnosti. Někteří naopak považují toto označení za přeceňované bezpečnostní opatření, které bylo v minulosti již tolikrát prohlášeno za lehce napadnutelné a prolomené, že implementace nepřináší žádnou přidanou hodnotu. Jistě se ale v dnešní době nedá mluvit o bezpečnosti a přitom nezmínit SSL/TLS. Znalost

[1]Společnost vydala až verzi SSL 2.0, verze 1.0 nebyla nikdy vydána.

této problematiky je neodmyslitelnou součástí pro vývoj aplikací, návrh zabezpečeného firemního prostředí nebo administraci systémů.

Integrace využívající webových aplikací přibývá a je pravděpodobné, že znalost problematiky šifrovaného webového spojení bude expandovat i do odvětví, která s informačními technologiemi přímo nesouvisí. Tento standard je však zajímavý i sám o sobě. Z hlediska konceptu kombinuje celou řadu technologií, které demonstrují své přednosti a nedostatky právě díky široce využívaným službám. Reálně implementované služby prověřují vhodnost dílčích komponent SSL/TLS velice důsledně prostřednictvím útoků, které zneužívají zjištěné zranitelnosti. Zranitelnosti neodmyslitelně patří ke komplexním systémům v informačních technologiích a jejich studování napomáhá lepší implementaci nových systémů, které se učí z chyb předchozích verzí. Řada nově nalezených zranitelností vychází z dříve nalezených chyb, které byly buď zapomenuty, nebo nekorektně opraveny.

Hlavním důvodem volby tohoto tématu byla především motivace pochopit problematiku HTTPS spojení do takové míry, aby bylo možné dále rozšířit znalost bezpečnosti počítačových systémů. Znalost problematiky na úrovní této práce by měla být v dnešní době považována za základ pro zájemce o počítačovou bezpečnost, což ovšem nebývá vždy pravidlem.

Práce je rozčleněna do několika částí. První část se zaměřuje na obecnou teorii, která je bezpodmínečně nutná pro pochopení navazujících kapitol. Proto by měl být dbán dostatečný důraz na její prostudování. Následující kapitoly práce se již zaměřují na specifické útoky zpracované do oddělených částí. Případná další teorie jednotlivých útoků je popsána vždy v teoretickém úvodu každé části. Následuje dokumentace k laboratorním úlohám zaměřená na specifický útok s detailními postupy realizace, přičemž se čtenář v praktické části vcítí do několika rolí. Nejprve bude problematika prezentována z pohledu útočníka, kde si čtenář vyzkouší identifikaci zkoumané zranitelnosti a následně bude jeho úkolem provést samotný útok. V třetí části se čtenář vcítí do role bezpečnostního experta, který má za úkol navrhnout nápravná opatření pro danou zranitelnost. Tento postup má tedy simulovat reálný způsob správy systému. Správce či bezpečnostní expert je nejprve obeznámen s teorií, následně ověřuje zranitelnost, která může být v některých situacích i testována kvůli prověření dopadu. Po určení dopadu je obvykle nutné navrhnout vhodná opatření snižující riziko způsobené zranitelností.

Každý popsaný útok má nadefinované hodnocení, které slouží k tomu, aby vzájemně lépe porovnalo celkové riziko útoku. Parametry útoku jsou přitom nadefinovány takto:

- Jméno útoku - jméno popisující útok v této práci.

- Obtížnost (o) - definuje ve stupnici 1–10 obtížnost realizace útoku, přitom 1 představuje nejnižší obtížnost.
- Časové nároky (n) - definují ve stupnici 1–10 čas nutný k realizaci útoku. Například se může jednat a odposlechnutí daného množství komunikace nebo časově náročné početní operace nutné k výpočtům. Přitom 1 představuje minimální časové nároky.
- Znalost (z) - definuje ve stupnici 1–10 znalost nutnou pro realizaci útoku. Přitom 1 představuje minimální požadavky na znalost problematiky.
- Dopad (d) - definuje ve stupnici 1–10 dopad v případě útoku, přičemž útok nemusí mít v některých případech tak fatální následky. Minimální dopad přitom představuje číslo 1.

Riziko r_1 je pak v této práci definováno jako:

$$r_1 = 10 \cdot \log \left[\frac{(11 - o) \cdot (11 - n) \cdot (11 - z) \cdot d}{1000} \right]$$

Výsledné upravené riziko r je pak dáno vztahem:

$$r = \begin{cases} r_1 < 1.28 & r = 1 \\ r_1 \geq 1.28 & 10 \cdot \log \lceil r_1 \rceil \end{cases}$$

Osobním zájmem autora práce je poskytnutí kvalitních studijních materiálů umožňujících studenty vhodně obeznámit s problematikou HTTPS útoků. Vybrané útoky jsou již ve většině případů ošetřeny. Z historického hlediska se při hledání nových útoků často vychází z již známých zranitelností, jejichž koncept lze zobecnit a aplikovat i v nových prostředích.

1 POPIS HTTPS

Tato kapitola pojednává o základech nutných pro detailní pochopení dalšího textu této práce. Pokud nebude uvedeno jinak, budu v této kapitole čerpat z knihy pana Oppligera [15].[1]

HTTPS komunikace je založena na bezpečnostních protokolech, které mají zajistit základní vlastnosti této komunikace, tedy autentičnost, důvěrnost a integritu. Autentičnost, tedy zajištění komunikace se správnou identitou, je obvykle zajištěna certifikátem. Důvěrnost je zajištěna obvykle symetrickou šifrou, která je nutná pro samotný přenost dat. Symetrická šifra se volí tak, aby odpovídala výkonu koncových zařízení a nárokům na bezpečnost. Integrita je řešena pomocí hashovacího algoritmu, který zajišťuje, že nedošlo ke změně dat.

Protokoly pro HTTPS komunikaci se postupem času měnily a upravovaly podle bezpečnostních a hardwarových požadavků. V devadesátých letech dvacátého století uvedla společnost Netscape Communications protokol SSL. Bezpečnost protokolu byla ovšem velice špatná, a to natolik, že protokol SSL 1.0 nebyl ani zveřejněn. Protokol SSL 2.0 byl zveřejněn, ale závažné bezpečnostní chyby byly nalezeny již po několika měsících. Protokol SSL 3.0 byl následně kompletně přepracován, zejména pak část, která řešila SSL handshake.

I přesto, že byl protokol SSL 3.0 dlouho používán, byl již zveřejněn a využíván protokol TLS 1.0, který je do značné míry využíván i dnes. Aktuální verze TLS protokolu, navazujícího na SSL, jsou verze TLS 1.1 a TLS 1.2, přičemž se v současné době pracuje na verzi TLS 1.3.

Protokoly SSL/TLS i jejich implementace v podobě OpenSSL a Secure Channel jsou velice často prověřovány na různé zranitelnosti. Každá nalezená zranitelnost představuje vysoký stupeň rizika, protože se protokoly používají zejména pro citlivá data.

Obecně se šifrovaná komunikace HTTPS považuje za bezpečnou, nemusí to ovšem platit vždy. Špatná konfigurace klientů či serverů může způsobit i na nejnovějších protokolech závažné bezpečnostní chyby.

1.1 SSL protokol

Tento předchůdce TLS protokolu se používá méně často, zejména kvůli nedostatečné bezpečnosti. Přesto se s ním lze setkat v systémech, u kterých je vyžadována zpětná kompatibilita, případně u hardwarových implementací bez možnosti dodatečného rozšíření vlastností.

[1] Problematika HTTPS je natolik komplexní, že v této kapitole budou uvedeny pouze stěžejní informace.

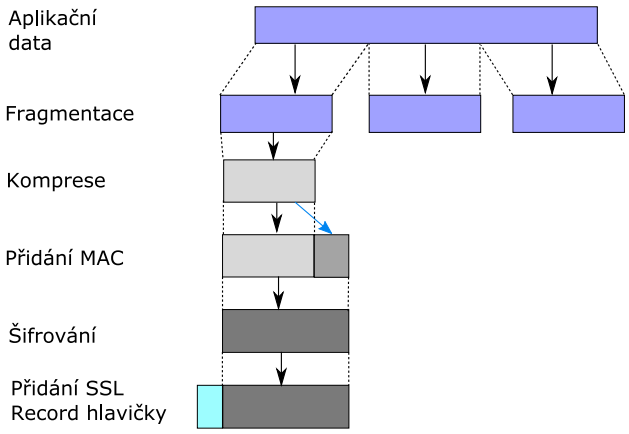
SSL protokol se dělí na následující části:

- Record protokol - slouží k rozdělení dat do bloků, kompresi dat, vytvoření MAC autentizačních dat pro zajištění integrity, zašifrování dat a vytvoření SSL záhlaví. SSL záhlaví obsahuje typ protokolu, verzi protokolu a celkovou délku dat.
- Hanshake protokol - slouží k zajištění výměny důležitých informací nutných pro šifrování.
- Change Cipher protokol - slouží ke změně šifrovacího algoritmu.
- Alert protokol - slouží k notifikaci o chybách při SSL komunikaci.
- Application data protokol - slouží ke vkládání dat do Record protokolu.

1.1.1 SSL Record protokol

Obrázek 1.1 popisuje, jakým způsobem se data zpracovávají. Po obdržení dat z aplikační vrstvy dochází k rozdělení do fragmentů, pak může volitelně následovat komprese. Z fragmentů se vypočítá MAC, která se přidá ke komprimovaným datům, následně dochází k šifrování a přidání záhlaví, které obsahuje:

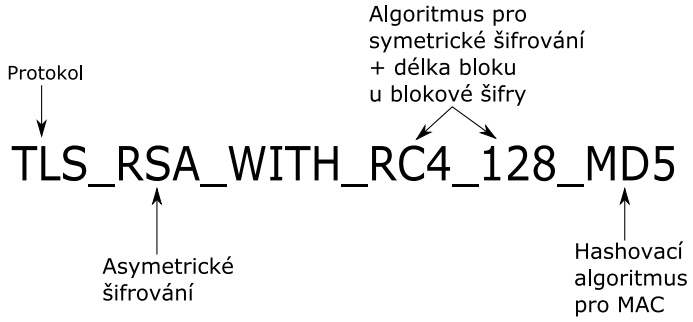
- Content Type (1 byte) - určuje typ zprávy.
- Major Version (1 byte) - určuje hlavní verzi protokolu.
- Minor Version (1 byte) - určuje podverzi protokolu.
- Compressed Lenght (2 byty) - určuje velikost komprimovaných dat ve zprávě.



Obr. 1.1: SSL Record protokol

Způsob šifrování se určuje pomocí tzv. cipher suite. Jedná se o kombinaci několika parametrů, přičemž pro označení dané kombinace algoritmů se využívá jejich textová interpretace.

Na obrázku 1.2 je popsána cipher suite, která pro výměnu klíčů využívá asymetrické šifrování RSA, pro symetrické šifrování potom RC4 s délkou klíče 128bitů, jako MAC využívá MD5.



Obr. 1.2: SSL cipher suite

1.1.2 SSL Handshake protokol

Je určen pomocí Content Type = 21, sjednává výměnu důležitých informací při sestavovaní spojení mezi klientem a serverem[2]. Obrázek 1.3 popisuje jednotlivé zprávy v čase.

Klient nejprve zasílá zprávu ClientHello, která obsahuje zejména podporované protokoly pro asymetrické šifrování, symetrické šifrování a MAC metody. Kombinace těchto tří parametrů se označuje jako cipher suite. Zpráva obsahuje také pole ID, které identifikuje relaci pro znovunavázání dřívějšího spojení. Pokud se jedná o novou relaci, ID obsahuje hodnotu 0. Klient ve zprávě volí, zda má být použita komprese, a současně posílá také náhodně vygenerované číslo „Client Random“.

Server odpovídá zprávou ServerHello, během které se posílají informace o vybraném cipher suite, certifikátu a ID spojení. Volitelně může odeslat také žádost o certifikát klienta a následně se posílá náhodně vygenerované číslo „Server Random“.

Server může poslat zprávu ServerKeyExchange, ta však není vždy nutná. Zpráva je nutná v případě, že se pro výměnu PMS klíče využívá metoda sestrojení klíče typu Deffie-Hellman a informace pro sestrojení nebyly součástí certifikátu serveru. Zpráva je nutná také v případě, že server využívá certifikát, který nepodporuje šifrování, ale pouze podepisování. V tom případě musí server vygenerovat dočasný RSA veřejný klíč, který pošle v této zprávě klientovi.

Server může také požádat klienta o certifikát pomocí zprávy CertificateRequest, pokud to bylo požadováno. Server dokončuje tento sled zprávou ServerHelloDone. V tuto chvíli má klient potřebné informace k tomu, aby byl schopný adekvátně odpovědět serveru.

V případě, že server poslal žádost o certifikát klienta, klient musí odpovědět zprávou Certificate. Klient dále odpovídá zprávou ClientKeyExchange. Tato

[2]Vzhledem k tomu, že se handshake ve verzi SSL 2.0 a SSL 3.0 liší, bude popis zaměřen na v současnosti využívanější verzi SSL 3.0.

zpráva je velice důležitá. Obsahuje informace o tajném klíči PMS zašifrovaném veřejným klíčem získaným z certifikátu serveru, jedná se o formát PKCS#1.

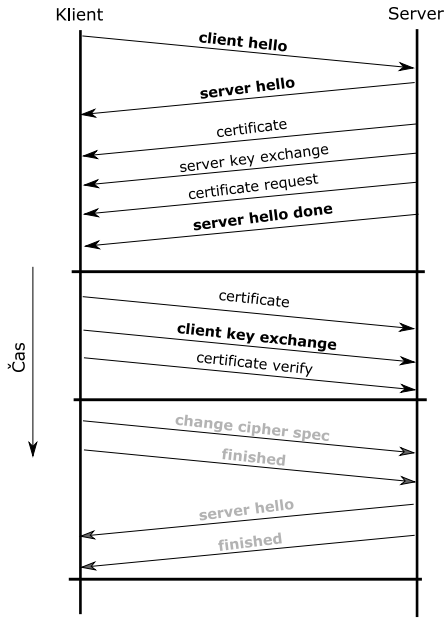
Pokud byl vyjednán algoritmus, který využívá sestrojení klíče, např. Deffie-Hellman algoritmus, pak se namísto zašifrovaného klíče posílají veřejné parametry pro sestrojení PMS. V případě použití pevně zvolených veřejných parametrů dle certifikátu tato zpráva není nutná.

Jestliže klient poslal serveru svůj certifikát, server musí poslat CertificateVerify, který obsahuje podepsaný hash zpráv vytvořených ze všech dosud poslaných zpráv podepsaných soukromým klíčem klienta.

Klient dále posílá zprávy ChangeCipherSpec a Finished, server odpovídá také dvojicí zpráv ChangeCipherSpec a Finished.

Tímto je handshake dokončen a obě strany mohou využívat symetrické klíče odvozené z MS. Důležité je podotknout, že zpráva ChangeCipherSpec nespadá do Handshake protokolu, ale jedná se o protokol SSL Change Cipher Spec. Tato zpráva není šifrovaná.

Zpráva Finished je již zašifrovaná a obsahuje kontrolní součet ze všech zpráv, které byly poslány během procesu vyjednávání. Tím je zamezena modifikace zpráv, např. změna cipher suite. Pokud by nedocházelo k tomuto prověření, útočník by mohl záměrně měnit šifrovací algoritmy a snižovat tak zabezpečení mezi klientem a serverem.



Obr. 1.3: SSL Handshake protokol

1.1.3 SSL Change Cipher Spec protokol

Jedná se o jednoduchý protokol, který obsahuje jen jednu zprávu, a to ChangeCipherSpec, která není šifrovaná vyjednávaným algoritmem. Zpráva opět obsahuje záhlaví Record protokolu, Content Type = 20. Tato zpráva upozorňuje na stav vyjednávání. Vzhledem k tomu, že při vyjednávání nedošlo k chybě, server musí být připraven, že další komunikace bude šifrovaná. Ihned následuje zpráva Finished, která se posílá v samostatném paketu, protože je již šifrovaná. Je to první zpráva, která je zašifrovaná vyjednaným algoritmem.

1.1.4 SSL Alert protokol

Tento protokol notifikuje o chybách při komunikaci nebo vyjednávání SSL kanálu, Content Type = 21. Datová část obsahuje dva byty, první nabývá dvou hodnot:

- 1 - warning, jde o notifikační zprávu o varování.
- 2 - fatal, došlo k závažné chybě.

V dalším bytu je číselná hodnota odpovídající konkrétní situaci a chybě, přičemž jednotlivé kódy chyb nejsou pro další popis podstatné. pro podrobnější seznámení lze doporučit RFC [7].

1.1.5 SSL Application Data protokol

Jak již bylo naznačeno, Application Data protokol slouží k předávání dat mezi aplikační a SSL vrstvou. Tento protokol komunikuje s Record protokolem a předává mu nebo z něj odebírá data.

1.2 TLS 1.0 protokol

TLS protokol vychází z SSL 3.0, protokol TLS 1.0 tak lze považovat za verzi SSL 3.1. Zásadní změny jsou ovšem patrné v generování MS a způsobu, jak pracuje PRF, tedy generování a odvození klíčů pro komunikaci.

Struktura protokolů TLS je shodná s SSL. Opět jsou zde dvě vrstvy. V nižší vrstvě je TLS Record protokol, který má identický formát záhlaví, přičemž např. verze protokolu TLS 1.0 je označena jako Major version 3, Minor version 1.

Číselná označení Content Type protokolů vyšší podvrstvy jsou identická s SSL, tedy:

- 20 - TLS Change Cipher Spec protokol.
- 21 - TLS Alert protokol.
- 22 - TLS Handshake protokol.
- 23 - TLS Application Data protokol.

V TLS 1.0 byly oproti SSL 3.0 odstraněny některé cipher suite. Je také důležité zmínit, že se změnilo textové označení cipher suite, které začínají s předponou TLS_ namísto SSL_. Pořadí a význam ostatních částí zůstal nezměněn. Další změnou byl způsob generování MAC, který byl v SSL odlišný od HMAC. V TLS se plně převzal formát HMAC [12].

TLS také změnil způsob zpracovávání certifikátů, kdy již není požadováno ověřování až k nejvyšší certifikační autoritě, ale stačí ověření certifikační autority, která daný certifikát vydala.

TLS zredukovalo počet podporovaných typů certifikátů na čtyři. Podporuje pouze:

- 1 - RSA podepisování.
- 2 - DSA podepisování.
- 3 - RSA podepisování s pevnou Diffie-Hellman vyměnou klíčů.
- 4 - DSA podepisování s pevnou Diffie-Hellman výměnou klíčů.

TLS 1.0 Alert Message protokol má oproti SSL 3.0 změněné kódy pro chybové hodnoty. Další výrazný posun je též ve zjednodušení konstrukce zprávy CertificateVerify, kdy je ze zprávy v handshake vytvořen dle použitého algoritmu hash a následně je zašifrován veřejným klíčem certifikátu. Změnil se také způsob vytváření zprávy Finished.

Od SSL 3.0 se proces realizace hanshake nezměnil, proto lze považovat schéma z části o protokolu SSL za aktuální i pro TLS 1.0.

1.3 TLS 1.1 a 1.2 protokol

Verze 1.1 opravila některé vlastnosti blokových šifer, např. namísto CBC začala využívat mód s ECB, a to kvůli zabránění CBC útokům. Jako další opatření proti CBC útokům upravila chování chybových odpovědí.

Vlastnosti TLS protokolů byly delegovány institutu IANA [8], aby byla zlepšena flexibilita schvalování změn. Při drobných úpravách tak nemusí docházet ke změnám ve specifikacích protokolu.

Verze 1.2 začala využívat jednodušší algoritmus pro hashování, namísto kombinace MD5 a SHA-1 se použil pouze jeden algoritmus. Dále bylo upraveno chování serveru ohledně podpory nových rozšíření. V této verzi došlo k lepší podpoře virtualizace, zejména k podpoře modelu, kdy webové servery s TLS nemusí mít vlastní specifickou IP adresu.

Klient si může definovat maximální velikost TLS fragmentu, což dříve nebylo možné, a dále může namísto klientského certifikátu poslat pouze adresu, kde je certifikát ke stažení. Také má možnost informovat server o tom, který z certifikátů je pro klienta důvěryhodný. Server tedy poskytne pouze certifikáty důvěryhodné

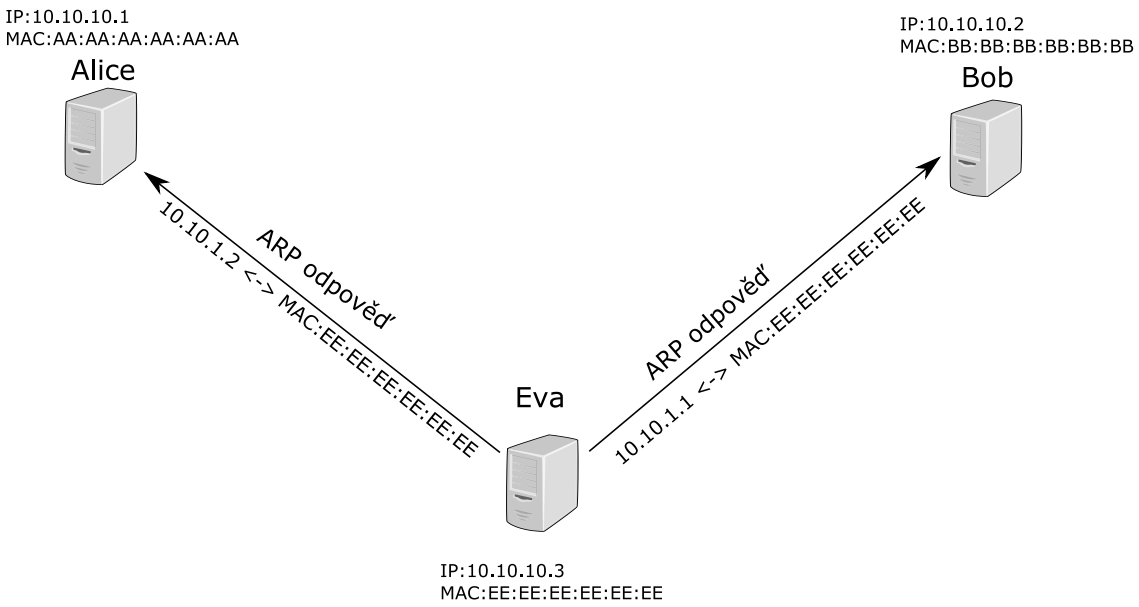
a nemusí docházet k vícenásobnému handshake ze strany klienta. Verze 1.2 též podporuje nové cipher suite, ať už se jedná o ECC, ECH nebo o AES. Rozšířila se také podpora jiných typů certifikátů pomocí rozšířených voleb.

2 DODATEČNÉ TEORETICKÉ PODKLADY

2.1 Útok ARP Spoof

Díky absenci autentizace u ARP může docházet k snadné manipulaci s informacemi o MAC adresách na všech zařízeních v dané broadcast doméně. Nekorektním zasíláním ARP odpovědí na neexistující dotazy dochází k záplavám chybných informací, které si počítače interpretují nežádoucím způsobem.

Komunikace pak nemusí být zasílána požadovanému zařízení, dokonce může být zasílána útočníkovi, který se vydává za cílové zařízení. Způsob realizace je znázorněn na obrázku 2.1.



Obr. 2.1: Popis ARP spoof

Počítač Eva posílá ARP odpověď počítači Alice, aniž by odpovědi předcházel dotaz. Tato odpověď informuje počítač Alice o špatné asociaci MAC adresy s IP adresou. Ve chvíli, kdy počítač Alice pošle ARP dotaz, získá ARP odpověď, kterou bude považovat za autentickou a upraví si svou ARP tabulku.

Podobným způsobem reaguje na falešné odpovědi i počítač Bob. Následkem toho jsou všechny rámce od počítače Alice zasílány počítači Eva, namísto počítači Bob. Podobná situace nastává i při posílání zpráv z počítače Bob počítači Alice. Pokud se zajistí přeposílání rámců tak, aby docházelo k doručení na počítače Alice a Bob, vše se zdá být v pořádku.

Počítač Eva se tak dostal do pozice, kdy monitoruje veškerou komunikaci mezi počítači, a této výhody může využít k případnému útoku. Počítač Eva je v pozici MITM.

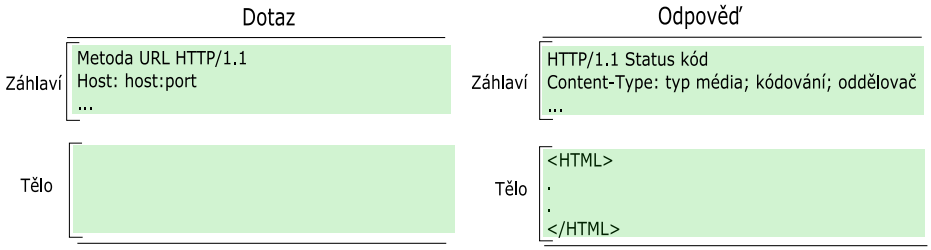
2.2 Znalost HTTP komunikace

Jako výchozí port pro komunikaci protokolu HTTP se využívá TCP/IP port 80, ale mohou být použity i jiné porty. Ve většině implementací HTTP/1.0 se využívalo nové připojení pro každý dotaz/odpověď. V HTTP/1.1 lze použít během jednoho připojení více operací dotaz/odpověď. Pro předávání proměnných cílovému serveru se využívá metod GET a POST.

Metoda GET slouží k obdržení informací o požadované stránce, např. data, která by měla být vrácena klientovi, data jsou viditelná v URI žádosti.

Metoda POST slouží k odesílání dat serveru, přitom data nejsou v URI adrese. Slouží obvykle k dodatečným informacím o otevřeném zdroji, zasílání zpráv, novinek a jiných textových dat nebo k posílání bloku dat a rozšiřování již existujících dat. Data v metodě POST jsou obvykle vázána na aktuálně používanou URI.

Zpráva je obvykle dělena na záhlaví a tělo zprávy, viz obrázek 2.2. Záhlaví HTTP zprávy obsahuje textové záznamy oddělené pomocí odřádkování CR a LF. Ukončení záhlaví je realizováno pomocí dvou po sobě jdoucích odřádkování. Po záhlaví následuje tělo HTTP zprávy. Více informací o HTTP správě lze zjistit z RFC dokumentace [5].



Obr. 2.2: HTTP záhlaví a tělo

Záhlaví umožňuje využívat pole X-Frame, které má za úkol zvýšení zabezpečení oproti zranitelnosti tzv. Clickjacking. Nejde však o specifikace a nemusí být dodržováno. Informace s tímto prefixem jsou obvykle ignorovány [20].

3 ÚTOK POODLE

Jméno útoku	Obtížnost	Časové nároky	Znalost	Dopad	Celkové riziko
Poodle	6	3	5	7	4

Realizaci útoku lze považovat za obtížnější. Útočník musí být nejen MITM, ale navíc musí být schopný uživateli podstrčit vhodný Java skript, který generuje nutné modifikované požadavky. Samotný MITM nemusí být realizován pomocí ARP spoof, je však nutné nějakým způsobem zajistit dohled na síťovým provozem HTTPS. Útočník musí mít také dobrou znalost šifrování SSL 3.0.

Při úspěšném útoku lze získat informace jako jsou hesla, popř. cookie, využívající se při autentizaci. Dešifrování komunikace je však časově náročné a existuje zde realné riziko, že platnost cookie skončí dřív, než ji útočník získá. Pokud nebude uvedeno jinak, kapitola bude čerpat z dokumentu [14].

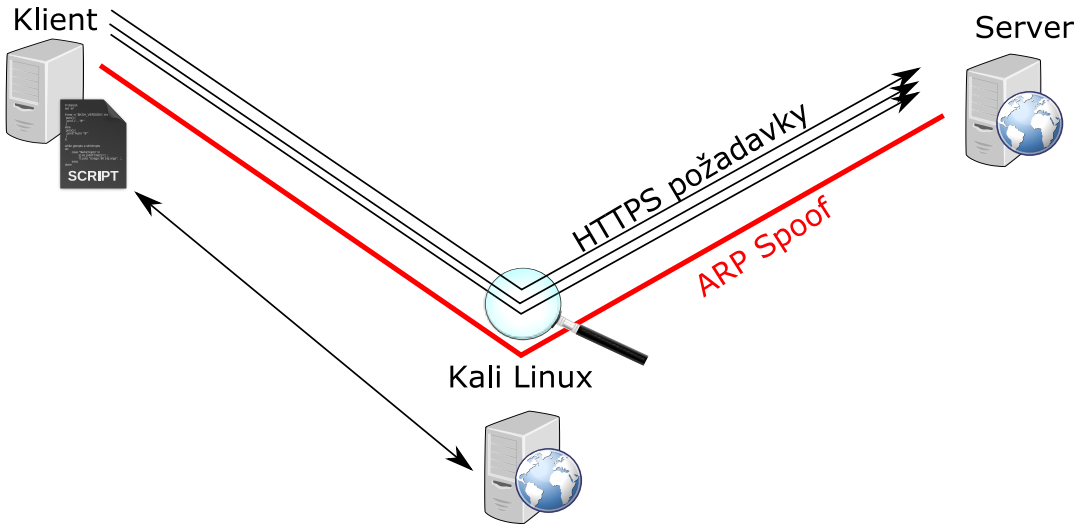
3.1 Teoretický úvod

POODLE útok, tedy „Padding Oracle On Downgraded Legacy Encryption“ popsal v roce 2014 Bodo Möller, Thai Duong a Krzysztof Kotowicz. Samotný útok byl podobný v té době již známému útoku BEAST. Využívá konceptu Oracle pro zajištění informací o posledním bytu bloku, na který se provádí útok.

Ke zjištění hodnoty jednoho bytu je nutné prověřit přibližně 2^8 možností, které jsou zasílány na HTTPS server, což z hlediska výpočetního výkonu není zásadní problém. Zranitelnost, kterou POODLE využíval, byla natolik zásadní, že přiměla mnoho společností úplně zakázat podporu SSL 3.0.

V dnešní době je protokol SSL 3.0 považován za nebezpečný a i přesto, že běžné webové prohlížeče už toto verzi protokolu běžně nepodporují, servery SSL 3.0 mohou podporovat. Zranitelnost lze využít i při útoku na jiné komunikační protokoly využívající šifrování SSL 3.0 jako je IMAP. U aplikací s přímou integrací podpory SSL 3.0 v kódu dochází k opravení podobných zranitelností poměrně pomalu nebo vůbec, pokud nejsou již podporovány dodavatelem. Proto nelze tuto zranitelnost popřít.

3.1.1 Topologie sítě laboratorní úlohy



Obr. 3.1: Topologie sítě pro útok POODLE

3.1.2 Vytvoření PKCS#7 v SSL 3.0 s CBC

HTTP požadavky šifrované blokovou šifrou jsou v SSL 3.0 rozdělené na stejně dlouhé fragmenty, přičemž délka fragmentu odpovídá typu blokové šifry. Požadavek tedy může být zpracován např. 64 bitovou (8 bytovou) šifrou typu 3DES. Takový případ je uveden na obrázku 3.2, kde je naznačeno dělení do jednotlivých fragmentů. V případě použití EAS by byla zpráva rozdělena do bloků o délce 16 bytů [1].

Dalším důležitým poznatkem je formát PKCS#7, který je popsán v příslušném RFC [11]. I přesto, že RFC doporučuje vkládat výplň v přesně daném formátu, implementace SSL 3.0 běžně využívá mírně pozměněný způsob záznamu.

RFC definuje výplň tak, že je ke zprávě připojeno n bytů, přitom každý připojený byte obsahuje hodnotu $n-1$. Velikost zprávy představuje proměnná l, velikost bloků určuje k. Pro výpočet n platí:

$$n = k - (l \bmod k) - 1 \tag{3.1}$$

GET /HT	TP/1.1\r\n	Cookie:	abcdefgh	\r\n\r\nXXXX	MAC data	******7

Obr. 3.2: Formát zprávy v PKCS#7

[1] EAS je 128bitová bloková šifra.

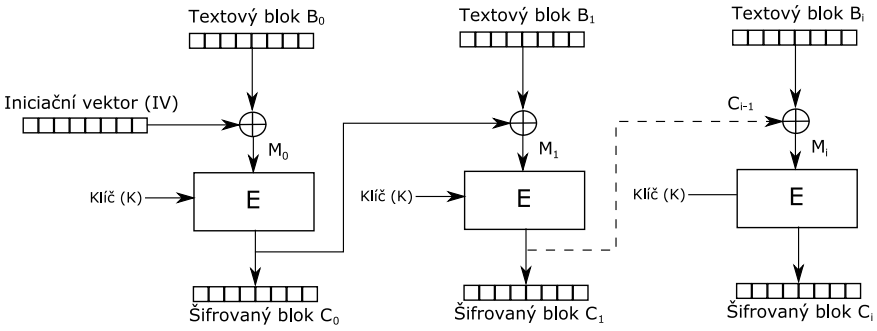
3.1.3 CBC

Předpokladem k pochopení tohoto útoku je nutné popsat způsob šifrování pomocí blokových šifer označovaných jako CBC. Proces šifrování znázorňuje obrázek 3.3. Data jsou rozdělena do bloků B_0 až B_n, přičemž na první blok je aplikováno XOR s inicializačním vektorem IV. Blok je následně zašifrován pomocí zvolené symetrické šifry. Výsledný zašifrovaný text E_0 je použit pro šifrování dalšího bloku. Proces šifrování lze popsat takto:

$$\begin{aligned} M_0 &= B_0 \oplus IV, \\ C_0 &= E(M_0). \end{aligned}$$

Pro další bloky pak:

$$\begin{aligned} M_i &= B_i \oplus C_{i-1}, \\ C_i &= E(M_i). \end{aligned}$$



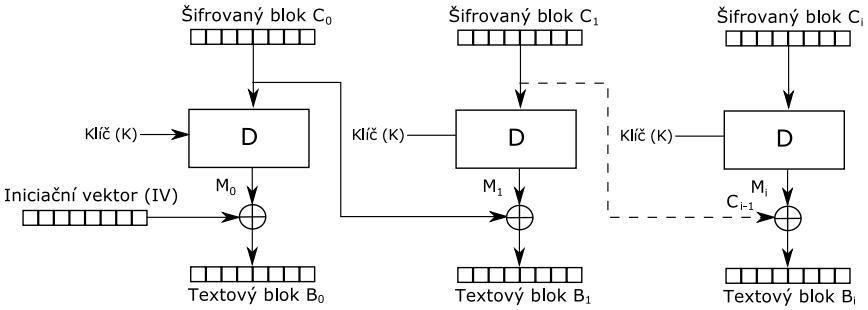
Obr. 3.3: Popis šifrování CBC

Dešifrování pracuje podobným způsobem, viz obrázek 3.4. První zašifrovaný blok C_0 se dešifruje pomocí modulu D. Modul D má na vstupu také klíč K, který byl použit při šifrování. Na výstup je aplikován XOR s inicializačním vektorem, výstupem je pak dešifrovaný text B_0. Proces dešifrování lze popsat takto:

$$\begin{aligned} M_0 &= D(C_0) \\ B_0 &= M_0 \oplus IV, \end{aligned}$$

Pro další bloky pak:

$$\begin{aligned} M_i &= D(C_i) \\ B_i &= M_i \oplus C_{i-1}, \end{aligned}$$



Obr. 3.4: Popis dešifrování CBC

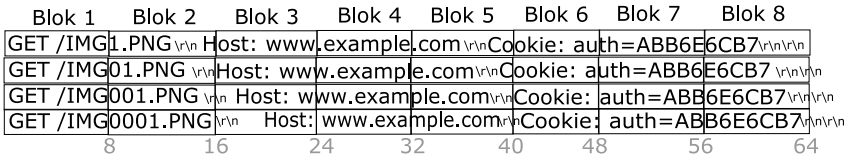
3.1.4 Princip útoku Poodle

Útočník musí zajistit několik důležitých předpokladů pro realizaci útoku. Nejprve musí útočník realizovat MITM, např. využítím ARP spoof. Tím je zajištěno, že je veškerá komunikace přesměrována na jeho počítač. Útočník následně spouští vhodně upravený proxy server, pomocí kterého může server provádět potřebné změny na datech posílaných k HTTPS serveru. Proxy server při vyjednávání propouští pouze komunikaci protokolu SSL 3.0. Tím se zajistí využití zranitelnosti.

Po sestavení spojení oběti se serverem oběť ověřuje identitu na serveru pomocí cookie, kterou má ve svém prohlížeči. Útočník na jiný webový server vložil vhodný skript, ten může být skrytý např. v reklamním okně. Útočník může využít i dalších technik či zranitelnosti ke spuštění skriptu, jako je spam s vloženým skriptem, nebo injekce kódu do webových stránek. Skript bude následně používán ke generování vhodných požadavků na serveru ze strany oběti a ke komunikaci s útočníkem. Útočník tímto komunikačním kanálem koordinuje délku a formát HTTPS požadavků.

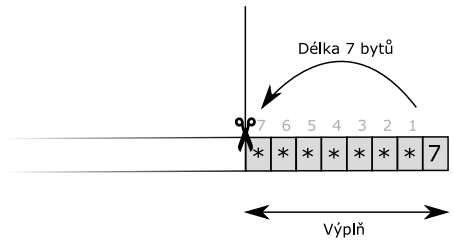
Jednou z klíčových informací pro útočníka je určení hodnoty posledního bytu zprávy, tedy určení výplně. Hodnotu posledního bytu nelze zjistit přímo, může ale modifikovat délku požadavků posílaných skriptem, který bude postupně zvětšovat požadavek o jeden byte, například přidáváním znaku do požadavku ve jméně souboru. Podobně jako je tomu znázorněno na obrázku 3.5.

Při zvětšování GET požadavků o $i \leq k$ dojde po určitém počtu nárůstů ke zvětšení celé zprávy o dalších n, tedy 8 bytů(v případě 3DES šifry). V tu chvíli je



Obr. 3.5: Modifikace požadavku

možné odvodit hodnotu $k = n - 1$, protože výplň je ve zprávě vždy vyžadována. Pokud dojde k navýšení o n bytů, je ke zprávě připojeno přesně 8 bytů (v případě 3DES šifry) výplně a poslední byte proto musí obsahovat hodnotu k, viz obrázek 3.6. Takto je určena hodnota výplně s posledním bytem zprávy a lze tak začít provádět útok. V tuto chvíli je tedy útočník obeznámen s hodnotou jednoho byte ve zprávě a na základě této znalosti může útočník útok provádět.



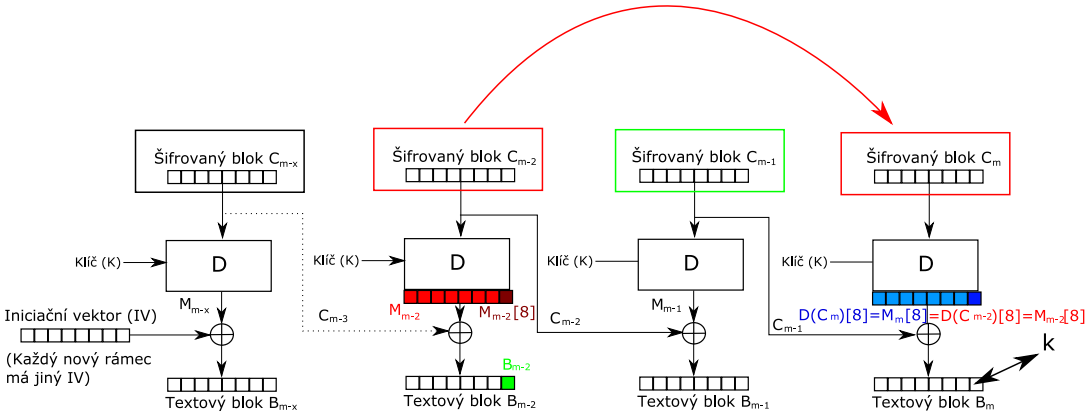
Obr. 3.6: Zjištění velikosti výplně

Než bude popsán další krok útoku, je nutné popsat proces ověření MAC. Ten lze rozdělit do následujících kroků:

- Dešifrování všech dat zprávy.
- Načtení posledního bytu, aby bylo možné zjistit, kolik bytů tvoří výplň.
- Odstranění bytů výplně.
- Načtení $MAC1$ a odpojení $MAC1$ ze zprávy.
- Výpočet nové $MAC2$, následné porovnání hodnot $MAC1$ s $MAC2$.

Tento proces tedy chrání před modifikací obsahu zpráv třetí stranou, změna obsahu by způsobila neshodu $MAC1$ a $MAC2$. Na první pohled lze považovat postup za důvěryhodný. Problémem však je, že data vložená do výplně jsou ignorována při výpočtu MAC.

Jestliže se na začátek zprávy vloží tolik bytů, aby výplň byla velká přesně n bytů, poslední byte bude obsahovat $n - 1$. Nejen že bude jasný obsah posledního bytu, navíc jsou všechny informace ve zbytku bytu ignorovány při výpočtu MAC.



Obr. 3.7: Dešifrování bloku dat

Poté, co je identifikovaný poslední byte zprávy, lze provádět přesun jiného bloku, např. C_{m-2} na pozici C_m, kde m představuje index posledního bloku. Po dešifrování posledního bytu v posledním bloku server vypočítá umístění MAC. Pokud byl poslední byte shodný s hodnotou původního bloku, dojde ke správnému zarovnání a shodnému výpočtu MAC. Ke správnému zarovnání většinou nedojde a server odesílá zdroji informaci o neshodě v MAC. Pokud by však byl poslední blok bytu dešifrován tak, že odpovídá hodnotě 7, došlo by k úspěšnému ověření MAC.

Každý nový požadavek způsobí vygenerovaní nově inicializačního vektoru IV. Nová hodnota IV způsobí při dodržení stejného, výše uvedeného, postupu dešifrování posledního bytu jiným způsobem a výsledná hodnota posledního bytu v přesunutém bloku by měla být jiná. Vzhledem k velké diverzitě lze předpokládat, že po přibližně 256 pokusech dojde k situaci, kdy není spojení po odeslání zprávy ukončeno, poslední byte ve zprávě je tedy 7. V takovém případě je možné provést dešifrování posledního bytu v šifrovaném bloku C_{m-2}, což je znázorněno na obrázku 3.7. Pak by se jednalo o dešifrování bloku $m-2$, tento index bloku lze zobecnit. Pro zobecnění bude použit znak j, což odpovídá pořadí bloku od začátku zprávy, který má být dešifrován. V případě že přesunutí bloku B_j na místo bloku B_m nezpůsobí chybu MAC, pak platí:

$$\begin{aligned} D(C_m[8]) &= D(C_j[8]) = M_m[8] = M_j[8], \\ B_m[8] &= k = C_{m-1}[8] \oplus D(C_m[8]), \\ M_j[8] &= k \oplus C_{m-1}[8], \\ B_j[8] &= C_{j-1}[8] \oplus M_j[8]. \end{aligned} \tag{3.2}$$

V tuto chvíli je jasně určen jeden (poslední) byte v bloku j, kde se může nacházet cookie. Následně se postupuje stejným způsobem, posun bytů pro cookie v rámci bloků se provede přidáním bytu na začátek zprávy a odebráním bytu na konci zprávy.

Tento proces je znázorněn na obrázku 3.8. Postupným posunováním obsahu se mění i byte, který byl dešifrován.

```
GET /a COOKIE dataabcdefgh výplň=7
GET /aa COOKIE dataabcdefg výplň=7
GET /aaa COOKIE dataabcdef výplň=7
GET /aaaa COOKIE dataabcde výplň=7
GET /aaaaa COOKIE dataabcd výplň=7
GET /aaaaaa COOKIE dataabc výplň=7
```

Obr. 3.8: Posun cookie v rámci zprávy

3.2 Praktická část

Tab. 3.1: Seznam počítačů a IP adresy

Jméno počítače	IP adresa
cl1	192.168.64.137
evil1	192.168.64.136
srv1	192.168.64.135

V praktické části bude demonstrován postup analýzy zranitelnosti POODLE a bude realizován samotný útok. Následně bude řešen způsob zabezpečení serveru. Pro realizaci zranitelnosti je nutné mít k dispozici HTTPS server s touto zranitelností. V této laboratorní úloze bude využita distribuce obsahující záměrně velké množství zranitelností. Distribuce bWAPP je volně k dispozici [9].

Spusťte virtualní stroje:

- **cl1**
- **evil1**
- **srv1**.

V případě sytémů **cl1** a **evil1** použijte účet **root** a heslo **toor**. V případě systému **srv1** použijte účet **bee** a heso **bug**.

3.2.1 Ověření zranitelnosti

Pro ověření zranitelnosti je možné využít různé programy, které ověřují servery i na další případné bezpečnostní problémy. Pro identifikaci zranitelnosti na útok POODLE postačuje program **Nmap** s využitím skriptu **ssl-enum-ciphers**. Tento

skript vypíše podporované cipher suite cílového serveru. Pro ověření zranitelnosti se připojte k počítači **evil1**, spusťte nmap s příslušnými parametry a na základě teoretických poznatků identifikujte problematické cipher suite.

Na virtuálním stroji **evil1** spusťte příkaz:
nmap --script ssl-enum-ciphers -p 443 srv1.vutbr.cz.

Otázka č. 1 - Identifikujte alespoň 4 cipher suite v kombinaci s verzí SSL/TLS, které způsobují zranitelnost POODLE.

3.2.2 První útok

V prvním útoku bude cílový server realizován pomocí programu **Socat**. Důvodem využití programu **Socat** je podrobný výpis stavu serveru. Díky tomu lze lépe prověřit vyhodnocení odpovědí od HTTPS serveru. **Socat** program bude spuštěn na cílovém serveru na portu **4443** a bude zpracovávat HTTPS požadavky, provoz bude přesměrován na port **80**, kde běží již existující HTTP služba.

Program **Socat** a certifikáty jsou k dispozici ve složce **POODLE** uložené na ploše, pro spuštění je tedy nutné spustit program přímo ze složky **POODLE**.[2]

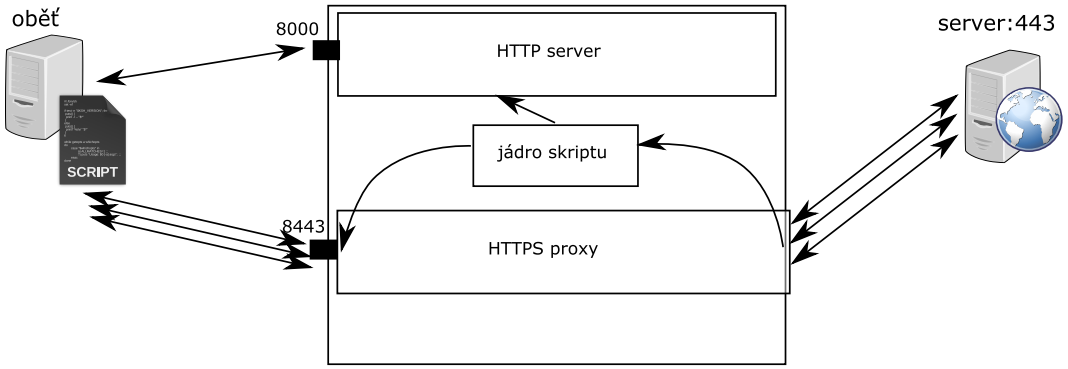
Na virtuálním stroji **srv1** ve složce **POODLE** spusťte příkaz:
socat -v -x
OPENSSL-LISTEN:4443,verify=0,method=SSLv3,cert=cert-poodle.pem,key=key-poodle.pem,reuseaddr,fork TCP:localhost:80.

Nyní je konfigurace na straně serveru hotova. Dostupnost služby na portu **4443** můžete prověřit pomocí prohlížeče přímo na počítači **srv1**.

K realizaci útoku bude využit skript, který publikoval Thomas Patzke [16]. Tento skript umožňuje flexibilní nastavení tak, aby útok mohl být spuštěn dle požadavků. Skript spouští dvě služby na počítači útočníka. Jedna HTTP služba běží na portu 8080 a slouží pro komunikaci s Java skriptem běžícím u oběti. Druhá HTTPS služba běží ve výchozím nastavení na portu **8443**, k tomuto portu se připojuje útočník a posílá zde HTTPS požadavky.

[2]Přepnutí do složky lze realizovat pomocí příkazu **cd Desktop\POODLE**.

Tyto požadavky se následně přesměrovávají na cílový HTTPS server a skript zpracovává odpovědi. Pomocí těchto odpovědí skript donastavuje informace posílané klientovi prostřednictvím portu 8000. Odpovědi cílového HTTPS serveru jsou v průběhu útoku převážně zprávy o chybě způsobené špatným kontrolním součtem MAC. HTTP server posílá informace o počtu vkládaných bytů na začátku a na konci zprávy. Tyto informace přijímá Java skript běžící na straně oběti a upravuje tak požadavky posílané přes HTTPS. Obrázek 3.9 popisuje princip tohoto skriptu.



Obr. 3.9: Princip skriptu poodle.py

Poodle skript spusťte tak, aby naslouchal na portu **8443**. Tento provoz bude následně posílán na cílový server, který naslouchá na portu **4443**. HTTP server, který běží na počítači útočníka, ponechejte na portu **8000**.[3] Skript je nutné spustit pomocí programu **Python3**. Pro spuštění skriptu je nutné zadat explicitně specifickou verzi interpretu. Skript lze spustit jednoduše pomocí příkazu **python3** ***jméno_python_skriptu.py*** s příslušnými parametry. Pokud spustíte skript bez udání parametrů, skript vypíše pomocný výpis všech paramterů.

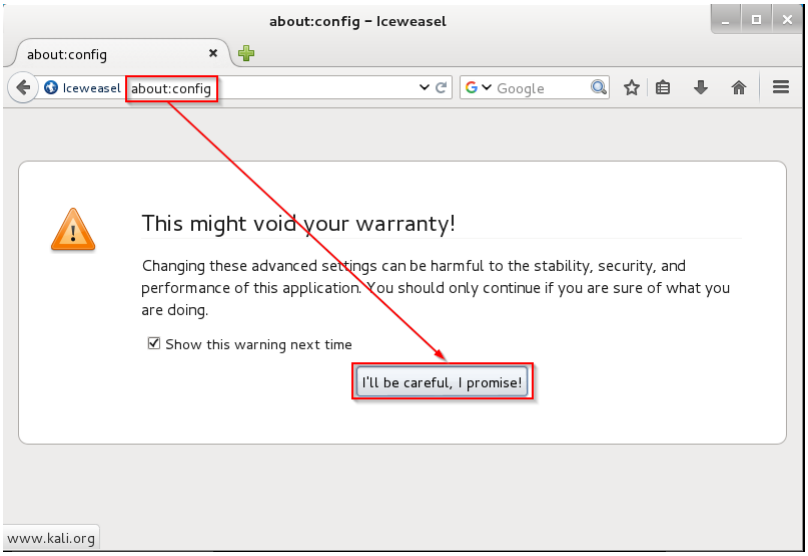
Na virtuálním stroji **evil1** ve složce **POODLE** spusťte **poodle.py** skript s tímto nastavením:

- offset nastavte na hodnotu **430**
- cílový port nastavte na hodnotu **4443**
- cílový host nastavte na hodnotu **192.168.64.135**
- uveďte parametr, který určuje URL adresu cílového server z pohledu oběti:
  https://192.168.64.136:8443/bWAPP/portal.php.

Pro správnou funkci útoku je nutné nakonfigurovat klientský počítač tak, aby podporoval a využíval pouze SSL 3.0 protokol. Konfiguraci prohlížeče na počí-

[3]Pro zobrazení všech voleb stačí spustit skript bez parametrů.

tači **cl1** prověříte spuštěním prohlížeče **Iceweasel**. Do adresního řádku zadejte **about:config** a potvrďte modifikaci nastavení prohlížeče, viz obrázek 3.10 .



Obr. 3.10: Konfigurace prohlížeče

Po otevření konfigurace prohlížeče lze nastavovat velké množství možností, přičemž pro snadnější hledání požadovaných voleb lze zadat do vyhledávacího okna požadované klíčové slovo pro nastavení.

Do okna **Search** zadejte „security.tls.version“ a ověřte tato nastavení:

- **security.tls.version.max = 0**
- **security.tls.version.min = 0**.

Pro zajištění vhodného výběru cipher suite je nutné zakázat ty, které nevyužívají CBC. Stačí vyhledat hodnoty „ssl3“, které obsahují text „rc4“, a ty následně vypnout.

Do okna **Search** zadejte „security.ssl3.*_rc4_*“ a ověřte nastavení všech klíčů na **false**.

Jako další krok je nutné zajistit načtení cookie do prohlížeče na počítači **cl1**, což lze zajistit přihlášením z tohoto počítače. Ve chvíli, kdy se pokusíte otevřít stránku prostřednictvím IP adresy serveru útočníka, dochází k chybě způsobené neshodnou adresou v certifikátu se jménem serveru, ke kterému se připojujete. Počítač **evil1** nemá legitimní jméno odpovídající certifikátu. V rámci testování můžeme tento fakt

ignorovat a výjimku certifikátu potvrďte. Následně se přihlaste na webové rozhraní pomocí účtu **bee** a hesla **bug**. Heslo v prohlížeči potvrďte a ponechejte heslo uložené.

Na počítači **cl1** spusťte prohlížeč a přihlaste se na stránku:
https://192.168.64.136:8443/bWAPP/portal.php.

Pro zajištění funkčnosti ponechejte okno prohlížeče otevřené a spusťte novou záložku v prohlížeči. Útok bude spuštěn otevřením stránky HTTP serveru na počítači útočníka, viz obrázek 3.11. V okně prohlížeče se objeví stránka, která informuje o spuštění požadavků generovaných z počítače **cl1** na cílový server prostřednictvím HTTPS proxy serveru běžícího na počítači **evil1**.

Na počítači **cl1** spusťte v nové záložce prohlížeče stránku:
192.168.64.136:8000.

Obr. 3.11: Spuštění útoku

Ověřte okno skriptu spuštěného na počítači **evil1** a okno programu **Socat** na počítači **srv1**. Pročtěte výpis obou oken a s vyučujícím diskutujte o tomto výpisu. Identifikujte alespoň 7 prvních bytů dešifrovaných skriptem.

Otázka č. 2 - Identifikujte alespoň 7 prvních bytů, které byly dešifrovány během útoku, a tyto byty/znaky zapište do protokolu.

Otázka č. 3 - Vysvětlete důvod zpráv „decryption failed or bad record mac“ v okně programu Socat.

3.2.3 Druhý útok

V druhém útoku využijte stejný skript, pouze místo **Socat** serveru využijte přímo HTTPS server běžící na serveru **srv1** na portu **443**. Pro změnu konfigurace stačí ukončit program **Socat**, uzavřít prohlížeč na počítači **cl1** a ukončit skript **Poodle** spuštěný na počítači **evil1**.

Ukončete běh programu **Socat** na serveru **srv1**, ukončete skript poodle na **evil1** a ukončete prohlížeč na počítači **cl1**.

Ověřte, že na serveru **srv1** běží HTTPS služba na portu **443**. Tato služba bude při útoku využívána jako cílová.

Na virtuálním stroji **evil1** ve složce **POODLE** spusťte poodle skript s tímto nastavením:

- offset nastavte na hodnotu **430**
- cílový port nastavte na hodnotu **443**
- cílový host nastavte na hodnotu **192.168.64.135**
- uveďte parametr, který určuje URL adresu cílového server z pohledu oběti:
  https://192.168.64.136:8443/bWAPP/portal.php.

Podobně jako v prvním útoku je nutné spustit na počítači **cl1** HTTPS stránku běžící na počítači **evil1** na portu **8443** a ve druhé záložce spusťit HTTP stránku běžící na počítači **evil1** na portu **8000**. Postupujte při konfiguraci počítače **cl1** stejně jako při prvním útoku. Zaznamenejte dešifrování alespoň 5 bytů ze skriptu poodle, proces druhého útoku opakujte i s hodnotou offsetu **432**. Oba výstupy porovnejte.

Spusťte prohlížeč na počítači **cl1** stejně jako při prvním útoku. Následně celý útok opakujte i s hodnotou offsetu **432**. Výstupy ze skriptů porovnejte.

Otázka č. 4 - Jakým způsobem ovlivňuje dešifrovaný výstup hodnota parametru offset?

3.2.4 Třetí útok

Poslední útok již bude realizován tak, aby byl útok z pohledu oběti transparentní. Na straně serveru **srv1** již nebude potřeba provádět žádné další změny, útok bude realizován na HTTPS serveru běžícím na portu **443**.

Zásadní změnou bude spuštění ARP spoof na počítači **evil1**. Dále bude nutné spustit přesměrování na počítači **evil1** dat z portu **443** na port **8443**. Z pohledu klienta tak bude poodle skript naslouchat na identickém portu, jako naslouchá server **srv1**. ARP spoof následně zajistí podvržení IP adresy. Nejprve je nutné ukončit běžící druhý útok.

Ukončete skript poodle na **evil1** a ukončete prohlížeč na počítači **cl1**.

Na počítači **evil1** spusťte poodle skript tak, aby cílová adresa odpovídala jménu cílového serveru.

Na počítači **evil1** ve složce **POODLE** spusťte poodle skript s tímto nastavením:

- offset nastavte na hodnotu **430**
- cílový port nastavte na hodnotu **443**
- cílový host nastavte na hodnotu **192.168.64.135**
- uveďte parametr, který určuje URL adresu cílového server z pohledu oběti:
  https://srv1.vutbr.cz/bWAPP/portal.php.

Na počítači **evil1** spusťte přesměrování provozu a nastavte pravidlo pro přesměrování komunikace z portu **443** na port **8443, dále spusťte ARP spoof mezi cl1 a srv1.**

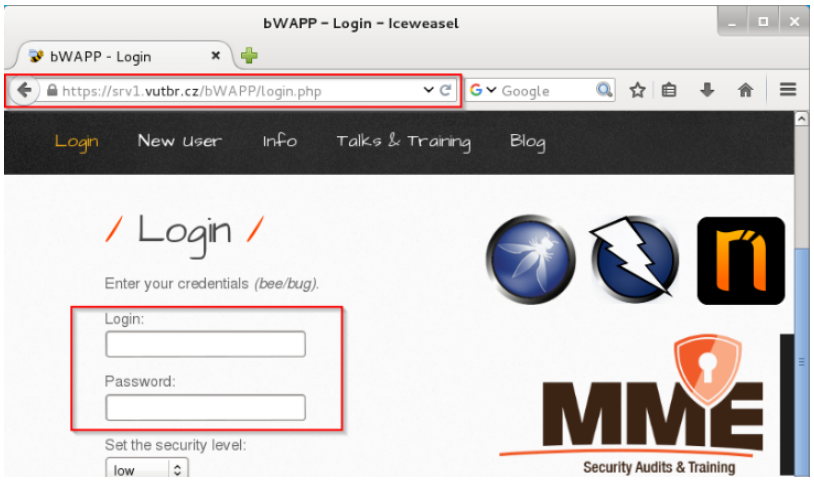
Povolení přesměrování a nastavení přesměrování portu lze provést pomocí těchto příkazů:

```
echo "1" > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A PREROUTING -p tcp --destination-port 443 -j
REDIRECT --to-port 8443.
```

Nastavení ARP spoof na počítači **evil1** lze provést tímto způsobem:

```
arpspoof -i eth0 -t 192.168.64.137 192.168.64.135
arpspoof -i eth0 -t 192.168.64.135 192.168.64.137
```

Na počítači **cl1** bude nutné přihlášení pod novou adresou, v tomto případě jde o adresu cílového serveru. Nyní již nedochází k chybě během přihlášení na server a přesto je komunikace díky ARP spoof přesměrována na počítač **evil1** na port **443**. Ten je následně přesměrován na port **8443**, kde je komunikace zpracována poodle skriptem a poslána na cílový server. Přihlášení na cílový server je znázorněno na obrázku 3.12.



Obr. 3.12: Spuštění útoku

Na počítači **cl1** spusťte v prohlížeči:

- na jedné záložce stránku:
  https://srv1.vutbr.cz/bWAPP/login.php a přihlaste se
- na druhé záložce stránku:
  192.168.64.136:8000.

V okně skriptu budou postupně dešifrovány jednotlivé byty, přitom oběť realizovala připojení bez jakékoliv chyby. Útok dešifruje byty ve zprávě od jiného znaku než při prvním a druhém útoku. Pokuste se identifikovat důvod této změny.

Otázka č. 5 - Proč došlo k dešifrování od jiného znaku než u prvního a druhého útoku?

Otázka č. 6 - Jakým způsobem by se docílilo dešifrování obsahu od stejného znaku jako pří prvním útoku?

3.2.5 Opatření proti zranitelnosti

Zamezení útoku POODLE je naznačeno jak v teoretické části, tak při zjišťování zranitelnosti. Pokuste se posoudit varianty zamezení této zranitelnosti. Nejde přitom o úpravu konkrétní konfigurace, pouze obecně naznačte nápravná opatření.

Otázka č. 7 - Na základě teoretických a praktických poznatků navrhněte možné způsoby zabezpečení serveru proti zranitelnosti POODLE.

4 ÚTOK TLS RENEGOTIATION

Jméno útoku	Obtížnost	Časové nároky	Znalost	Dopad	Celkové riziko
TLS Renegotiation	5	2	5	5	3

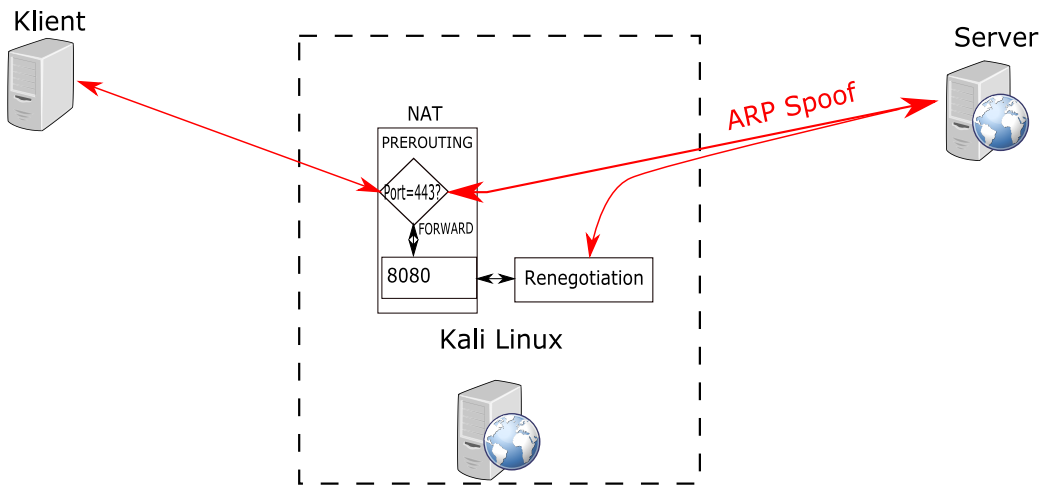
Jedná se o relativně obtížný útok, který ani neumožní získání cookies nebo hesla. Přesto lze docílit vyvolání změn, které budou provedeny v kontextu oběti. Přitom lze tento útok provést v krátkém čase.

4.1 Teoretický úvod

Útok TLS Renegotiation (dále jen „Renegotiation“)je efektivní útok, který umožňuje upravit žádost od oběti tak, aby odpovídala požadavkům útočníka. Útok byl publikován v roce 2009, autory jsou Marsh Ray a Steve Dispensa [17]. Pokud nebude uvedeno jinak, budu z tohoto zdroje čerpat informace obsažené v této kapitole.

Útok postihuje TLS 1.0 a SSL 3.0. Zranitelnost byla nezávislá na implementaci, útok je tedy možné realizovat jak na servery s OpenSSL, tak i na Microsoft systémy s IIS. Tato metoda neumožňuje přímý útok na samotný zašifrovaný kanál, umožňuje však narušení integrity dat [3].

4.1.1 Topologie sítě laboratorní úlohy

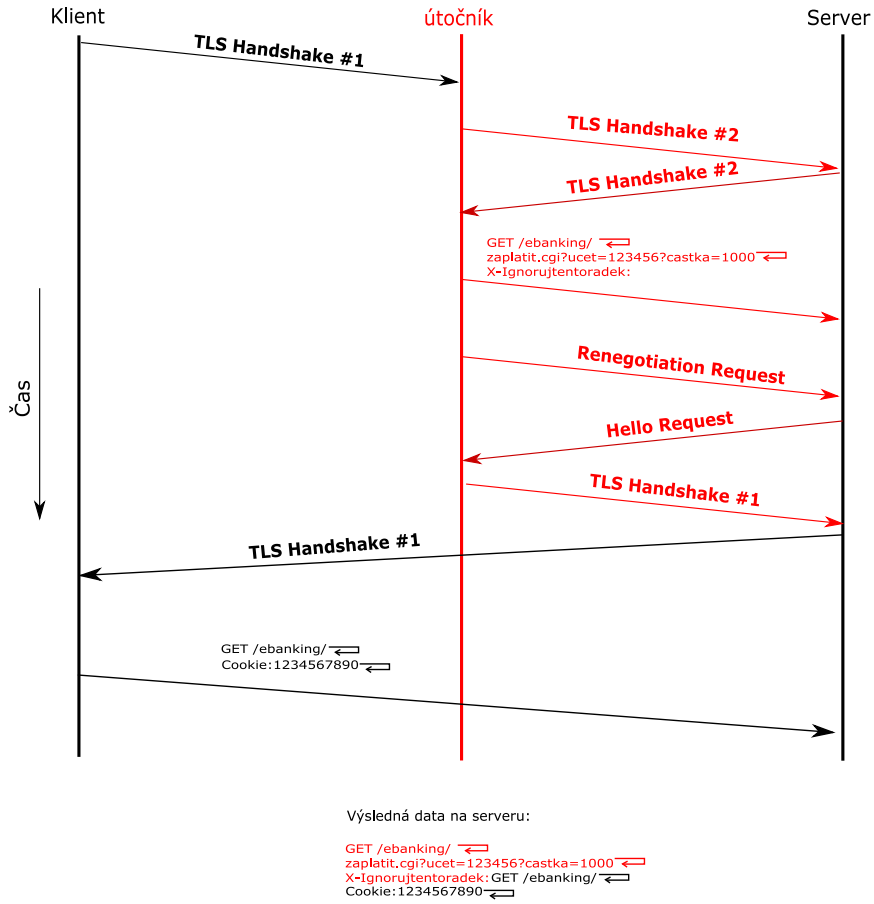


Obr. 4.1: Popis Renegotiation

4.1.2 Princip útoku

Útočník musí sestavit MITM pomocí ARP Spoof, přičemž veškerá komunikace mezi webovým serverem a obětí je směrována přes jeho počítač.

Celý útok popisuje obrázek 4.2. Oběť se pokouší sestavit TLS handshake s HTTPS serverem, útočník však zprávu ClientHello pozdrží a namísto ní provede TLS handshake s HTTPS serverem sám. ARP spoof zajistí podvržení klienta, proto se ze strany serveru zdá, že server komunikuje s klientem a ne s útočníkem.



Obr. 4.2: Popis útoku Renegotiation

Útočník má nyní zabezpečený šifrovaný kanál a server čeká na požadavky. Útočník posílá začátek požadavku na webový server, přičemž v požadavku posílá metody, které chce realizovat s oprávněním oběti.

Útočník může zprávu posílat postupně, podvrstvy posílají informace aplikaci až ve chvíli, kdy je celý požadavek přijat. Do té doby server stále čeká na dokončení HTTP zprávy. Na posledním řádku útočník posílá text „X-Ingoruje-tento-radek:“, server zprávy X-Frame nevyužívá a interpretuje si tento řádek tak, že následující data v tomto řádku budou ignorována. Útočník záměrně nepošle na konci žádosti

ukončení řádku CR a LF [1]. Server tedy očekává další komunikaci, přičemž je jasné, že první přijatý řádek bude interpretovat jako pokračování řádku, který má ignorovat. Následující řádky již přidá ke zprávě, kterou přijal dříve.

```
¨GET /bezpecne/index.html HTTP/1.1\r\n¨
¨Host: banka.com\r\n¨
¨Connection: keep-alive\r\n¨
¨\r\n¨
¨GET /prevod/transakce.php?ucet=123456789&castka=1000 HTTP/1.1\r\n¨
¨Host: banka.com\r\n¨
¨Connection: close\r\n¨
¨X-ignoruj-tento-radek: ¨
```

Obr. 4.3: Zpráva od útočníka

V tuto chvíli musí útočník zajistit spojení takto připravených dat čekajících v paměti serveru s daty klienta. Útočník posílá požadavek na změnu šifrování k serveru, server posílá zprávu HelloRequest a očekává zaslání nového ClientHello v již sestaveném zabezpečeném spojení.

Útočník zasílá ClientHello oběti, která iniciuje renegotiation proces. Tento požadavek posílá prostřednictvím svého vyjednaného zašifrovaného kanálu. Útočník od této chvíle pouze přeposílá veškerou další komunikaci klienta, protože již není schopný zasahovat do komunikace mezi klientem a serverem. To, že další sledování provozu není možné, ovšem nevadí. Útočník docílil zaslání požadovaných dat pod identitou klienta.

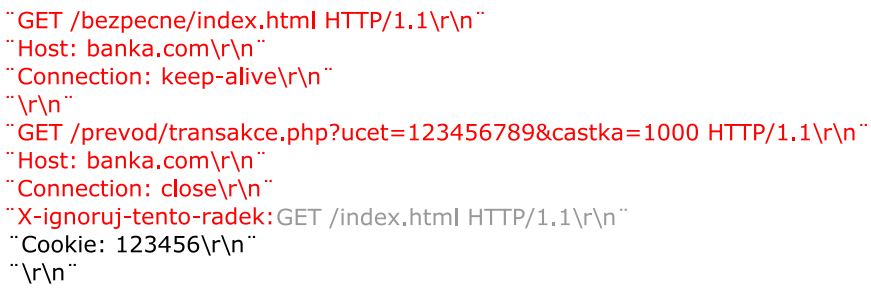
```
¨GET /index.html HTTP/1.1\r\n¨
¨Cookie: 123456\r\n¨
¨\r\n¨
```

Obr. 4.4: Zpráva od oběti

Ze strany klienta dochází k dokončení renegotiation procesu. Tento samotný proces neovlivňuje HTTP data čekající na zpracování. Z pohledu serveru nastává další aktivita příchodem zprávy GET od klienta. Jakmile oběť zašle zprávu, část s metodou GET je ignorována (tedy první řádek) a zbývající část celé zprávy, včetně cookie oběti, se připojí k datům poslaným útočníkem. Příklad zprávy je znázorněn na obrázku 4.4. Data se z pohledu služby spojí do jednoho požadavku a ten je zpracován bez toho, aniž by bylo zřejmé, že během posílání požadavku došlo k renegotiation. Zpráva z pohledu serveru je uvedena na obrázku 4.5.

To, že server může tuto zranitelnost obsahovat, ovšem nemusí znamenat, že může dojít ke zneužití. Případný útok může být odvrácen vhodnou implementací jednorázových hodnot generovaných pro každou novou relaci. Tyto hodnoty zajišťují

[1] Kombinace CR a LF představuje nový řádek v HTTP zprávě.



Obr. 4.5: Zpráva z pohledu serveru

unikátní data vložená v datové části zprávy. Může jít např. o data typu NONCE [6], která zajišťují unikátnost. Případné spojení více požadavků do jednoho způsobí nevhodnou změnu těchto dat. Podobný způsob chrání aplikace před řadou dalších zranitelností, které se pokouší podvrhnout data. Integrace takto upravených unikátních dat by měla být samozřejmostí všech bezpečných aplikací.

4.1.3 Program pro ověření zranitelnosti

K účelu útoku je zvolen server, který vykazuje tuto zranitelnost. Pro ověření lze použít skript **SSLyze**.

```
sslyze --regular ServerIP:Port
```

Po zvolení vhodného parametru se jménem a portem serveru se v sekci „Renegotiation“ zobrazí výstup „VULNERABLE - Secure renegotiation not supported“. Jednou z dalších možností prověření je použití programu **OpenSSL**, kdy je možné zranitelnost i prakticky vyzkoušet.

```
openssl s_client -connect ServerIP:Port
```

Pokud server nepodporuje bezpečný renegotiation, objeví se na výstupu informace „Secure Renegotiation IS NOT supported“. To ovšem neznamená, že je podporováno i nezabezpečené renegotiation, což lze ověřit dále popsaným způsobem. Ověření renegotiation lze simulovat zápisem začátku dotazu HTTP. Následné zadání znaku „R“ informuje **OpenSSL** cílový server o pokusu o renegotiation. Pokud se objeví výstup „RENEGOTIATING“ bez chybového ukončení spojení, server je zranitelný.Po spuštění programu **OpenSSL** program očekává znaky, které posílá cílovému serveru. Pro otestování zranitelnosti zadejte tento text, přitom dejte pozor na správný formát včetně mezer:

```
HEAD / HTTP/1.0
R
```

4.2 Praktická část

Tab. 4.1: Seznam počítačů a IP adresy

Jméno počítače	IP adresa
cl2	192.168.64.142
evil1	192.168.64.136
evil2	192.168.64.139
srv2	192.168.64.141

V praktické části bude demonstrován postup analýzy zranitelnosti Renegotiation a bude realizován samotný útok. Následně bude řešen způsob zabezpečení serveru.

Spusťte virtuální stroje:

- **cl2**
- **evil2**
- **srv4**

Útok Renegotiation bude demonstrován pomocí skriptu vydaného společností RedTeam jako proof of concept útoku [18]. Skript byl dále implementován s webovou aplikací zveřejněnou na této stránce [24]. Konfigurace aplikace spolu se skriptem byla implementována pomocí distribuce Backtrack 5R1, do které byl dodatečně nakonfigurován Python a příslušné moduly nutné pro spuštění skriptu. Prostředí modeluje situaci, kdy oběť objednává pizzu pomocí webové aplikace. Útočník přitom využije ověření oběti a pod její identitou provede objednávku vlastní pizzy na jinou adresu.

Nejprve se připojte k počítači **srv4**, Backtrack 5R1 automaticky nespouští grafické prostředí. Před spuštěním grafického rozhraní je nutné přihlášení k cílovému systému. Tato distribuce má na všech systémech laboratorní úlohy stejný účet pro přihlášení. Jméno účtu je **root**, heslo k účtu je **toor**. Po přihlášení spusťte grafické prostředí ručně pomocí příkazu **startx**.

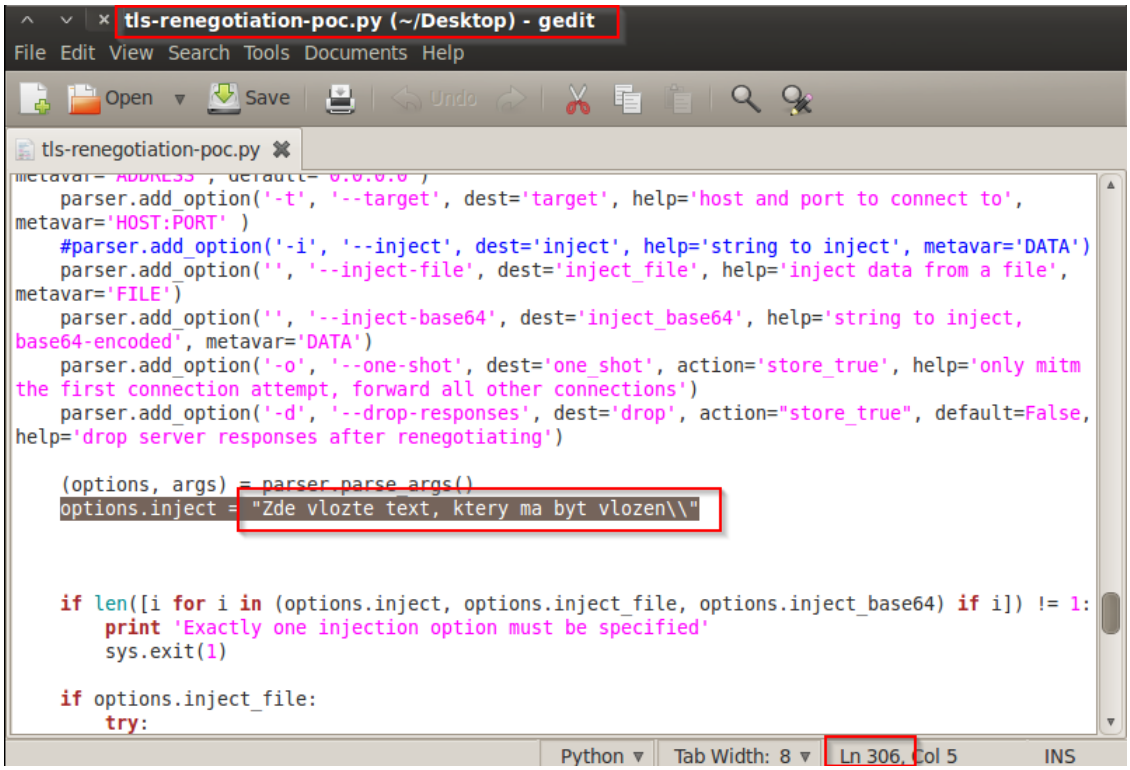
Ověřte dostupnost služby na počítači **srv4** spuštěním odkazu **PizzaTime** na ploše.

Pokud služba běží, připojte se na počítač **evil2** a ověřte zranitelnost pomocí programu **OpenSSL**, nebo **spusťte** počítač **evil1** a použijte pro ověření zranitelnosti skript **sslyze**. Výstup skriptu uložte pomocí snímání obrazovky. [2]

[2]Operační systém Windows umožňuje sejmutí obrazovky pomocí klávesy **Print Screen**. Ten

Otázka č. 1 - Využijte programu **OpenSSL**, nebo skriptu **sslyze** pro ověření zranitelnosti. Výstup zaznamenejte snímáním obrazovky. Následně vložte obrázek do protokolu.

Připojte se k počítači **evil2** a pomocí preferovaného editoru otevřete soubor **tls-renegotiation-poc.py**. Na řádku 306 má být vložen text, který bude reprezentovat požadavek posílaný metodou GET. Viz obrázek 4.6.



Obr. 4.6: Úprava skriptu

Tento požadavek bude obsahovat:

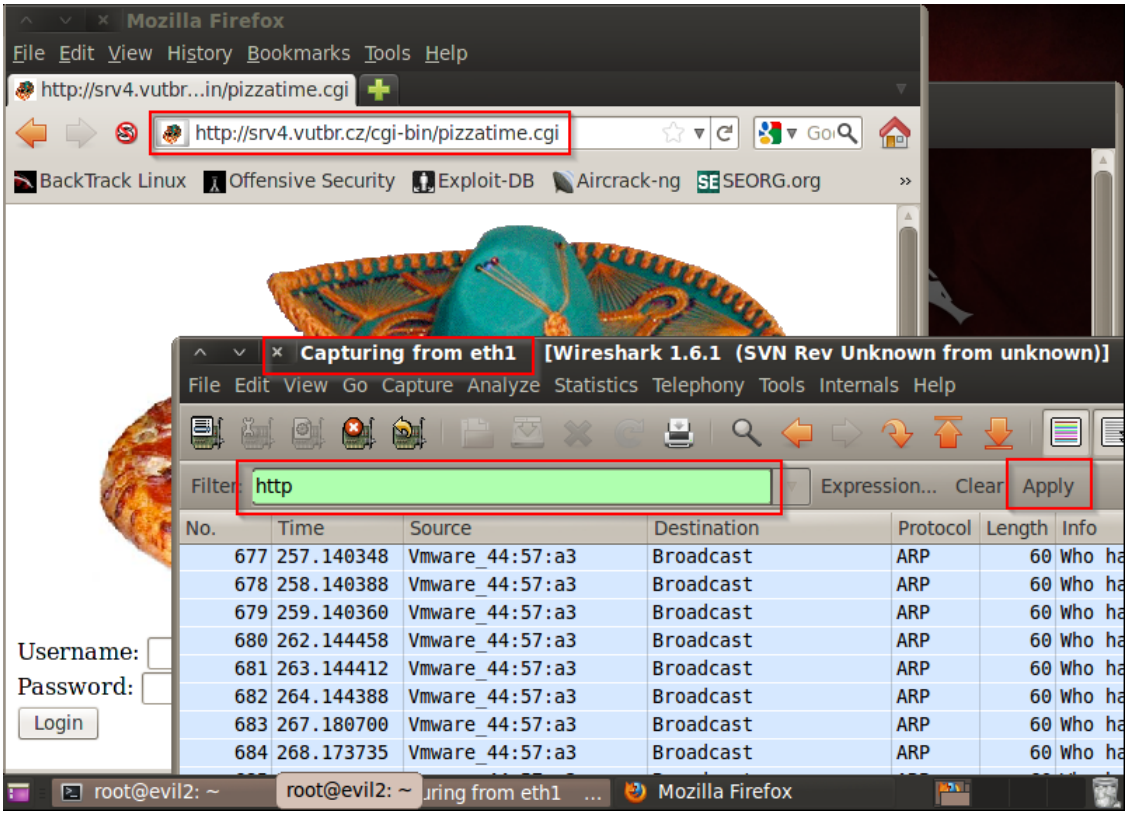
- **adresa** = adresa útočníka
- **pizza** = velká pizza Pepperoni

Správné upravení požadavku GET vyžaduje znalost formátu, která musí být nejprve zjištěna. Útoku tedy předchází analýza provozu a způsobu, jakým klientský počítač komunikuje s cílovým serverem. V našem případě je analýza usnadněna tím, že webová aplikace je dostupná také přes HTTP na portu 80. V reálných podmínkách by mohly být informace o způsobu komunikace zjištěny například pomocí přihlášení

uloží viditelnou oblast obrazovky do schránky. Následně lze tuto schránku vložit např. do programu **Malování**. Program **Malování** lze spustit pomocí příkazu **Mspaint.exe**.

jiným účtem a simulace akcí, které mají být realizovány pod účtem oběti. V tomto případě však aplikace obsahuje pouze jeden účet, proto stejný účet využijete k analýze formátu požadavků.

K analýze požadavků využijte **Wireshark** na počítači **evil2**. Program lze spustit z **terminálu** pomocí zadání příkazu **Wireshark**. Spusťte záznam provozu na síťovém rozhraní **eth1** a pomocí prohlížeče **Firefox** se připojte k serveru běžícímu na počítači **srv4**. Pro přihlášení využijte uživatelský účet **username** a heslo **password**. Pro přehlednost lze filtrovat provoz v programu **Wireshark** pomocí zadání filtru „http“, viz obrázek 4.7:



Obr. 4.7: Spuštění programu Wireshark

Na počítači **evil2**:

- Spusťte program **Wireshark** a spusťte záznam síťového provozu na síťovém rozhraní eth1
- Spusťte program **Firefox** a přihlaste se na adresu: **http://srv4.vutbr.cz/cgi-bin/pizzatime.cgi**.

Po úspěšném přihlášení do aplikace bude nutné provést objednávku. To je jeden z možných způsobů, jak zjistit formát zprávy, která se pro objednávku využívá.

Nejvýhodnější je vložení objednávky v požadovaném formátu, tím předejdete dalším nutným úpravám textu.

Na počítači **evil2** proveďte objednávku takovým způsobem, jaký je požadován v zadání.

V programu **Wireshark** identifikujte HTTP požadavek, kterým je provedena objednávka. Zprávu uložte pomocí snímání obrazovky a vložte obrázek do protokolu.

Otázka č. 2 - Zaznamenejte obsah zprávy požadavku pomocí snímání obrazovky. Následně obrázek vložte do protokolu.

Otázka č. 3 - Pokuste se najít další způsob, jak lze zjistit formát požadavku.Využijte program **Firefox**.

Otázka č. 4 - Pokuste se najít další způsob, jak lze zjistit formát požadavku, jiný než program **Firefox** nebo **Wireshark**.

Text vložte do kódu skriptu tak, aby nahradil text v uvozovkách. Přitom nahraďte konec textu „\r\n“ pomocí dvou zpětných lomítek podobně, jak tomu bylo u původního textu ve skriptu.

Na počítači **evil2** upravte skript tak, aby řádek **306** obsahoval požadavek útočníka.

V tuto chvíli je skript připraven k útoku. Předpokladem úspěšného provedení útoku je ale také přesměrování provozu z portu **443** na port, který využívá skript. Ten naslouchá na portu **8080**. Je nutné spustit přesměrování IP provozu s uvedenými parametry.

Na počítači **evil2** povolte přesměrování a nastavte přesměrování portu **443** na port **8080** a povolte ARP spoof mezi **cl2** a **srv4**.

Pro nastavení přesměrování použijte tyto příkazy:

```
echo "1" > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A PREROUTING -p tcp --destination-port 443 -j
REDIRECT --to-port 8080
```

```
arpspoof -i eth1 -t 192.168.64.141 192.168.64.142
arpspoof -i eth1 -t 192.168.64.142 192.168.64.141
```

Počítač **evil2** je připravený k útoku. Stačí spustit skript **tls-renegotiation-poc.py**. Skript je nutné spustit pomocí programu **Python2.7**. Pro spuštění skriptu je nutné zadat explicitně specifickou verzi interpretu. Skript lze spustit jednoduše pomocí příkazu **python2.7** ***jméno_python_skriptu.py***. Skript dále očekává několik parametrů. Specifikování IP adresy a portu cílového serveru se provádí pomocí parametru **-t**. Specifikace lokální IP adresy, pomocí které se provádí útok pomocí parametru **-b**. Specifikace lokálního portu, na kterém skript naslouchá, se provádí pomocí parametru **-l**.

Na počítači **evil2** ve složce **Desktop** spusťte renegotiation skript s tímto nastavením:

- cílová IP adresa s portem **192.168.64.141:443**
- lokální IP adresa **192.168.64.139**
- lokální port **8080**

Připojte se k počítači **cl2** a spusťte prohlížeč **Firefox**. Otevřete stránku serveru **srv4** a proveďte přihlášení k aplikaci pomocí uživatelského účtu **username** a hesla **password**. V okně prohlížeče přitom potvrďte uložení hesla do prohlížeče. **Pokud se objeví nový požadavek o přihlášení, tak přihlášení opakujte. Opakujte přihlašování, dokud neproběhne úspěšně.**

Na počítači **cl2** proveďte přihlášení k aplikaci na adrese:
https://srv4.vutbr.cz/cgi-bin/pizzatime.cgi.

Po úspěšném přihlášení zvolte adresu oběti a pizzu, kterou si bude chtít oběť objednat. Pro lepší přehlednost zvolte jiné údaje než ty, které byly vloženy do skriptu na počítači **evil2**.

Na počítači **cl2** se pokuste provést objednávku pizzy, kterou objednává oběť.

Po odeslání objednávky dojde k vložení požadavku zadaného útočníkem do skriptu. Prověřte si výstup aplikace a výstup spuštěného skriptu. Uložte snímek obrazovky skriptu a aplikace. Snímek obrazovky by měl být podobný snímku na obrázku 4.8.

Obr. 4.8: Výstup aplikace po útoku

Otázka č. 5 - Zaznamenejte obsah výstupu skriptu **tls-renegotiation-poc.py** a výstupu aplikace pomocí snímání obrazovky. Následně obrázky vložte do protokolu.

Otázka č. 6 - Jakým způsobem by se zachoval HTTP server, pokud by byl vložen text, který neodpovídá formátu HTTP?

4.2.1 Opatření proti zranitelnosti

Řešení zranitelnosti se liší v závislosti na platformě. Na platformě **OpenSSL** podléhají zranitelnosti verze 0.9.8k a dřívější. Pokud jde o novější verzi **OpenSSL**, ale přesto je útok Renegotiation možný. Pravděpodobně je to způsobeno nastavením

v konfiguraci webového serveru. U služby **Apache** je toto nastavení způsobeno volbou „SSLInsecureRenegotiation on“ v příslušném konfiguračním souboru. Nastavení je popsáno v dokumentaci [2].

Aplikace **Pizzatime** je spuštěna prostřednictvím HTTPS služby **lampp** na serveru **srv4**. Služba je nainstalována v lokaci **/opt/lampp**. Identifikujte důvod zranitelnosti. Jedná se o špatnou volbu v konfiguraci serveru **Apache**, nebo jde o zastaralou verzi **OpenSSL**? Zaměřte se na informace v souborech uložených v této složce a v podsložkách.

Na počítači **srv4** ověřte důvod zranitelnosti.

Otázka č. 7 - Identifikujte důvod zranitelnosti a navrhněte způsob řešení zranitelnosti. Opravu však neprovádějte, pouze popište možné kroky nápravy.

5 ÚTOK HEARTBLEED

Jméno útoku	Obtížnost	Časové nároky	Znalost	Dopad	Celkové riziko
Heartbleed	1	1	2	4	7

Heartbleed lze považovat za jednoduchý útok, který nevyžaduje velké časové nároky na realizaci. Rovněž nevyžaduje velkou znalost, ani MITM a lze provést prakticky odkudkoliv. Pokud tomu nebude uvedeno jinak, v této kapitole budou použity informace z tohoto zdroje [19].

5.1 Teoretický úvod

Tato zranitelnost postihuje HTTPS služby (SSL a TLS) konfigurované na OpenSSL verzi 1.0.1 až verzi 1.0.1f. Díky této chybě muže útočník získat blok dat o velikosti 64 KB dat, která mohou obsahovat citlivé informace jako soukromý klíč certifikátu nebo cookies.

5.1.1 Topologie sítě laboratorní úlohy

Obr. 5.1: Heartbleed útok - zapojení

5.1.2 Princip útoku

Podstatou této zranitelnosti je operace podobná přetečení zásobníku. Klient posílá na server informace pomocí metody Heartbeat a deklaruje velikost těchto informací. Server odpovídá a přikládá danou informaci v odpovědi.

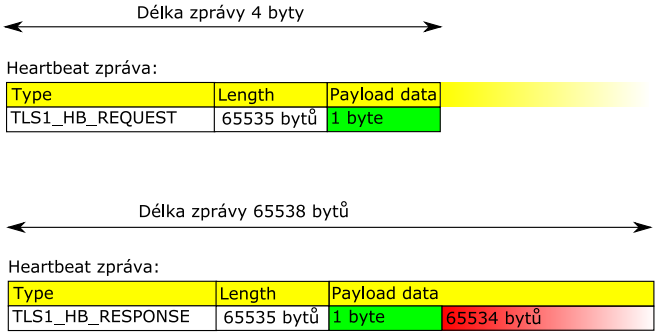
Toto rozšíření bylo implementováno do protokolu, aby byl klient schopný ověřit dostupnost serveru. Pokud však klient zašle serveru informaci menší, než je reálná velikost, ovšem deklaruje informaci o větší velikosti, server se spoléhá na deklarovanou velikost a odpoví klientovi zpět s přijatými informacemi, přitom ale z paměti načte blok dat odpovídající deklarované velikosti. Tyto dodatečné informace nemusí

být pro útočníka důležité, mohou však obsahovat také privátní klíč certifikátu nebo cookie, hesla k účtům a další senzitivní informace.

Velkou výhodou tohoto útoku je možnost opakování dotazu. Útočník tak postupně může získat více bloků o velikosti 64 kB a může tak získat velké množství podstatných informací.

Z pohledu bezpečnosti byla tato chyba velmi kritická. Pokud má server tuto zranitelnost, je nutné provést řadu zásadních kroků, které v některých případech nebyly provedeny. Tato zranitelnost teoreticky kompromituje veškeré informace, které může mít server v paměti. Další používání certifikátů umožňuje kompromitaci komunikace i na serverech, které tuto chybu nemají. Proto musí dojít ke zneplatnění všech certifikátů využitých na zranitelných serverech. Dále je nutné analyzovat veškeré informace, které na serveru byly, a změnit všechna hesla a zneplatnit cookies.

Obrázek 5.2 znázorňuje upravený dotaz, který deklaruje větší velikost, než zpráva skutečně má. Následná odpověď obsahuje 65534 bytů z paměti serveru.



Obr. 5.2: Heartbleed útok

5.2 Detailní popis útoku

Útok využívá rozšíření Heartbeat. Tato metoda zasílá požadavek na server a server odpovídá stejnou zprávou zpět klientovi. Klient posílá zprávu s těmito parametry:

- Content Type = 24, což označuje rozšířené volby.
- Type = „TLS1_HB_REQUEST“.
- Length = 65535.
- Data = „X“. Může se jednat o jakýkoliv 1 byte.

Server odpovídá zprávou s parametry:

- Content Type = 24.
- Type = „TLS1_HB_RESPONSE“.
- Length = 65535.
- Data = „X“ + 65534 bytů z paměti serveru.

5.2.1 Program pro ověření zranitelnosti

Zranitelnost lze prověřit celou řadou programů pro testování, které jsou k dispozici na distribuci Kali Linux. Tento popis obsahuje jen několik možností, jak zranitelnost prověřit.

Nmap

Program Nmap je k dispozici jak na Linux prostředí, tak na Windows. Program umožňuje načíst skripty, které jsou určeny pro konkrétní specifické testy. Pro testování zranitelnosti Heartbleed lze spustit příkaz:

```
nmap -sV --script=ssl-heartbleed ServerIp -p ServerPort
```

SSLyze

Jedná se o program, který je součástí Kali Linux. Lze jej použít pro řadu testů, pro test Heartbleed lze použít příkaz:

```
sslyze --heartbleed ServerIp:ServerPort
```

5.3 Praktická část

Tab. 5.1: Seznam počítačů a IP adresy

Jméno počítače	IP adresa
cl1	192.168.64.137
evil1	192.168.64.136
srv1	192.168.64.135

V praktické části bude demonstrován postup analýzy zranitelnosti Heartbleed a bude realizován samotný útok. Praktická část bude také demonstrovat možnost zneužití cookie k získání neprivilegovaného přístupu. Následně bude řešen způsob zabezpečení serveru. K samotnému útoku není nutné spouštět klientský počítač, nutné kroky lze realizovat přímo na počítači **srv1**.

Spusťte virtuální stroje:

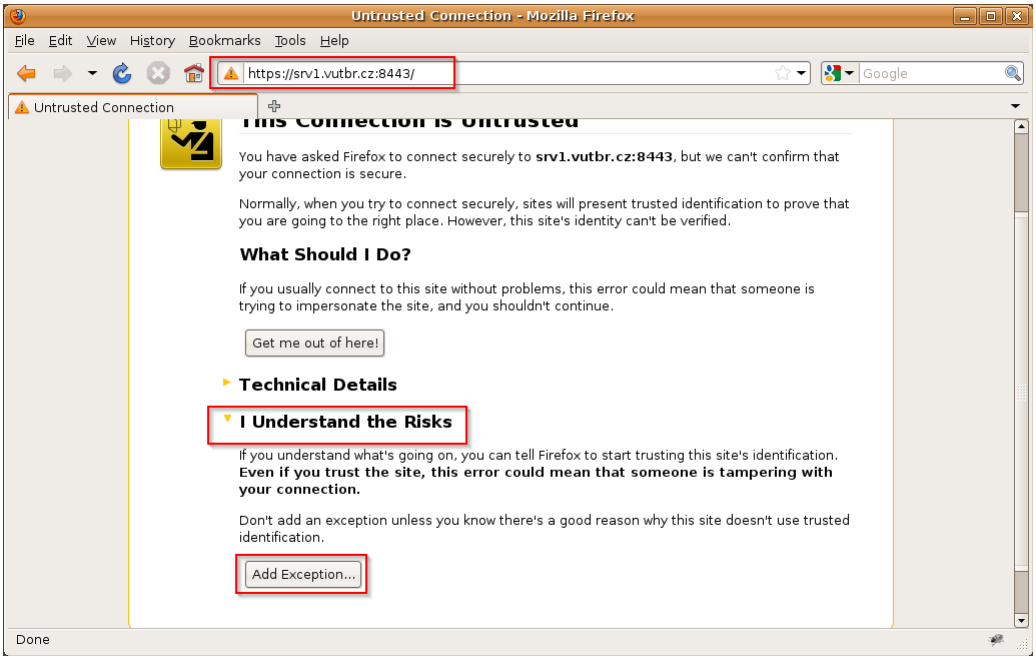
- **evil1**
- **srv1**

Útok Heartbleed zneužívá informací v paměti HTTPS serveru. K tomu, aby paměť serveru obsahovala zajímavé informace, je nutné tyto informace nejprve v paměti vygenerovat, konkrétně se jedná o cookie. Cookie představuje jedinečný identifikátor, kterým aplikace zamezí vícenásobnému ověření identity, uživatel tak nemusí zadávat přihlašovací údaje víckrát během jedné relace. Cookie rovněž reprezentuje sadu oprávnění pro autentizovanou identitu. Pro realizaci zranitelnosti je nutné mít k dispozici HTTPS server s touto zranitelností. V této laboratorní úloze bude využita distribuce obsahující záměrně velké množství zranitelností. Distribuce bWAPP je volně k dispozici [9].

Připojte se k serveru **srv1**. V případě, že je nutné přihlášení, využijte účet **bee** a heslo **bug**. Spusťte program **Firefox** a do adresního řádku zadejte HTTPS adresu serveru s cílovým portem **8443**. Server běžící na tomto portu obsahuje zranitelnost Heartbleed.

> Na serveru **srv1** spusťte program **Firefox** a do adresního řádku zadejte: **https://srv1.vutbr.cz:8443**

Vzhledem k tomu, že program **Firefox** nedůvěřuje certifikátu serveru, je nutné potvrdit důvěryhodnost. Postup potvrzení je znázorněn na obrázku 5.3 a 5.4.



Obr. 5.3: Povolení certifikátu

Poté, co je certifikát schválen, dojde k otevření HTTPS stránky. Následně je nutné vybrat položku **bWAPP** pro přihlášení k aplikaci. Při vyzvání k zadání účtu

Obr. 5.4: Potrvrzení certifikátu

zadejte účet **bee** a heslo **bug**. Pokud budete vyzváni k zapamatování hesla v prohlížeči, potvrďte uložení.

> Vyberte položku **bWAPP** a proveďte přihlášení na webovou aplikaci. Okno prohlížeče nezavírejte!

5.3.1 Ověření zranitelnosti

Před samotným útokem prověřte zranitelnost serveru, který naslouchá na portu 8443. K ověření zranitelnosti využijte poznatků z teoretické části. Zvolte program pro ověření zranitelnosti. Výstup uložte pomocí snímání obrazovky. [1] Na počítači **evil1** se připojte pomocí účtu **root** a hesla **toor**.

> Přihlaste se na počítač **evil1** a prověřte zranitelnost na cílovém serveru **srv1** pomocí jednoho z programů uvedených v teoretické části.

[1] Operační systém Windows umožňuje sejmutí obrazovky pomocí klávesy **Print Screen**. Ten uloží viditelnou oblast obrazovky do schránky. Následně lze tuto schránku vložit např. do programu **malování**. Program malování lze spustit pomocí příkazu **mspaint.exe**.

Otázka č. 1 - Zaznamenejte obsah výstupu programu na ověření zranitelnosti pomocí snímání obrazovky. Následně obrázek vložte do protokolu.

5.3.2 Realizace útoku

K realizaci útoku bude využit počítač **evil1** se skriptem **heartbleed.py**. Autorem skriptu je Travise Lee, tento skript je volně dostupný [13]. Ke spuštění skriptu je nutné spustit skript z terminálu v adresáři **Heartbleed** uloženém ve složce **Desktop** [2]. Skript je vytvořený v programovacím jazyku Python 2.7.

Pro spuštění skriptu je nutné zadat explicitně specifickou verzi interpretu. Skript lze spustit jednoduše pomocí příkazu **python2.7** ***jméno_python_skriptu.py***. Skript **heartbleed.py** vyžaduje zadání atributu jména cílového serveru a parametru specifikujícího číslo portu. Parametr portu je specifikován pomocí **-p** a následně číslem portu.

Na počítači **evil1** ve složce **Heartbleed** spusťte heartbleed skript s tímto nastavením:

- cílové jméno serveru **srv1.vutbr.cz**
- cílový port **8443**

Pokud byl útok úspěšný, zobrazí se informace „returned more data than it should - server is vulnerable!“. V případě neúspěšného útoku spusťte příkaz znovu.

Otázka č. 2 - Zaznamenejte obsah výstupu po úspěšném útoku skriptu **heartbleed.py** pomocí snímání obrazovky. Následně obrázek vložte do protokolu.

Otázka č. 3 - Promyslete si, co se přesně při útoku stalo. Zejména pak vztah mezi zprávou, která se posílá při útoku na server, a zprávou zasílanou zpět. Popište stručně, k čemu při útoku došlo a proč.

V případě, že dojde k úspěšnému vypsání paměti HTTPS serveru, výstup může obsahovat některé důležité informace, které mohou být využity při dalších útocích. Zejména se jedná o informace jako cookies, popř. hesla účtů, nebo soukromé klíče.

[2] Absolutní cesta ke složce je **/root/Desktop/Heartbleed**

Výstup skriptu analyzujte a identifikujte citlivé informace, které unikly při útoku ze serveru.

> Otázka č. 4 - Identifikujte ve výstupu skriptu citlivé informace, které unikly při útoku.

Ve výstupu skriptu bude také cookie, která byla vygenerována přístupy k serveru **srv1** v prohlížeči přímo na serveru. Pro ověření této skutečnosti otevřete v programu **Firefoxu** na serveru **srv1** add-on **Cookies Manager+**. V seznamu cookie, které bylo vygenerováno při přístupu na adresu **srv1.vutbr.cz**, najděte odpovídající cookie. Jméno cookie by mělo odpovídat jménu zobrazenému ve výstupu skriptu, tedy **PHPSESSID**. Porovnejte hodnoty cookie.

> Porovnejte hodnotu cookie **PHPSESSID** na výstupu skriptu a v prohlížeči, kde jste se přihlásili k aplikaci.

Na základě porovnání cookie je zřejmé, že informace obsažené ve výstupu skriptu mohou být využity pro ruční vytvoření cookie, pomocí které lze k serveru neoprávněně přistupovat pod jinou identitou. Pokud nedojde ke zjištění hesla přímo z výstupu skriptu, pořád lze provádět na systému akce, které mají oprávnění vlastníka daného cookie. Na počítači **evil1** vytvoříte ručně cookie v prohlížeči **Iceweasel**. Prohlížeč lze nalézt na hlavní liště, viz obrázek 5.5. Pomocí vytvořeného cookie se přihlásíte k HTTPS aplikaci na serveru **srv1** [3].

Obr. 5.5: Prohlížeč Iceweasel

> Na počítači **evil1** spusťte prohlížeč **Iceweasel** a otevřete add-on **Cookies Manger+**.

Kliknutím na **New Cookie** spusťte dialog pro ruční přidání cookie. Do pole **Name** přidejte jméno zachyceného cookie a do pole **Content** vložte unikátní identifikátor obsažený v cookie (32 znaků). Do pole **Domain** vložte plné jméno serveru včetně domény. Pole **Path** upravte tak, aby obsahovalo pouze lomítko. Uložte takto upravené cookie.

[3] Pro zobrazení menu v programu **Iceweasel** stiskněte klávesu **Alt**.

Na počítači **evil1** vytvořte v add-on **Cookies Manger+** cookie na základě informací z výstupu skriptu.

Nyní obsahuje prohlížeč cookie, které by mělo zajistit privilegovaný přístup k aplikaci běžící na cílovém serveru. Důležité je otevření cílového serveru pomocí dlouhého názvu. Vzhledem k tomu, že počítač útočníka nedůvěřuje certifikační autoritě, která podepsala certifikát pro server **srv1**, bude zobrazeno varování. Toto varování akceptujte.

Na počítači **evil1** se připojte prostřednictvím prohlížeče **Iceweasel** na adresu: **https://srv1.vutbr.cz:8443**
Potvrďte případné varování o nedůvěryhodné autoritě.

V prohlížeči se zobrazí výpis možných akcí, které lze na vzdáleném serveru provádět. Cookie, které bylo ručně vytvořeno, je určeno pro aplikaci bWAPP. Kliknutím na odkaz dojde k využití cookie a automatickému přihlášení k aplikaci bWAPP. Během zjišťování informací o klientovi aplikace nebude schopná identifikovat volbu **security level**, proto vyberte volbu **low** a potvrďte výběr tlačítkem pro nastavení, viz obrázek 5.6.

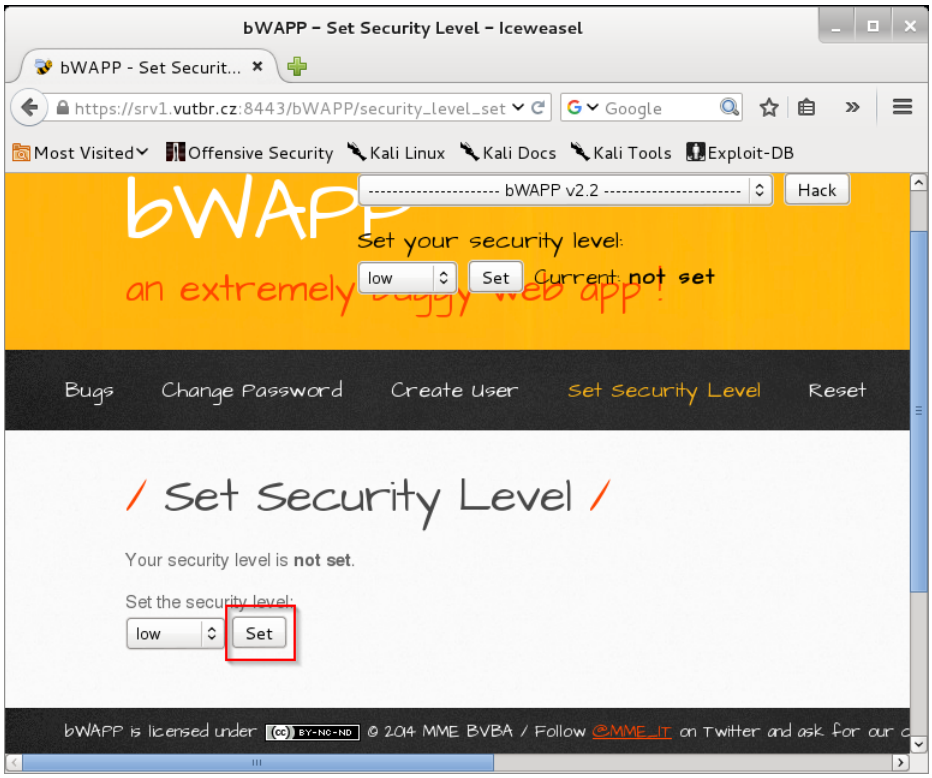
Na počítači **evil1** vytvořte v add-on **Cookies Manger+** cookie na základě informací z výstupu skriptu.

V tento okamžik máte přístup k aplikaci bWAPP jako uživatelský účet, který se původně přihlásil na lokálním serveru. Pro ověření funkčnosti přístupových práv vytvořte testovací účet. Ten lze vytvořit kliknutím na položku **Create User**. Během vytváření prosím zadejte parametry:

- Pole **Login** – zvolte přihlašovací údaje pro testovací účet.
- Pole **E-mail** – zvolte email testovacího účtu.
- Pole **Password** – zvolte heslo testovacího účtu.
- Pole **Re-type password** – zadejte opět heslo testovacího účtu.
- Pole **Secret** – zadejte text pro připomenutí hesla.

Na počítači **evil1** vytvořte nový účet v aplikaci.

Po vytvoření hesla obdržíte potvrzení „User sucessfully created!“. Pro ověření



Obr. 5.6: Nastavení security level

funkčnosti nově vytvořeného účtu se pomocí nového účtu přihlaste. Nejprve je nutné odhlásit se z účtu **Bee** kliknutím na položku **Logout** v pravé části lišty aplikace. Po přihlášení vytvořte snímek prokazující přihlášení pod novým účtem. Jméno účtu po přihlášení je patrné v pravé části lišty, kde je napsáno **Welcome**.

Otázka č. 5 - Zaznamenejte přihlášení pod nově vytvořeným účtem pomocí snímání obrazovky.

V případě dostatku času lze simulovat útok, kdy se pomocí počítače **cl1** oběť připojí prostřednictvím účtu, který jste právě vytvořili. Útočník z počítače **evil1** spustí **Heartbleed** skript a pokusí se získat informace, které by ho autorizovaly na cílový server. Pokuste se najít víc možností, jak na základě výpisu paměti daný účet kompromitovat.

Případně simulujte útok pomocí počítače **cl1**, pomocí kterého se připojte na server **srv1** nově vytvořeným účtem. Tento účet kompromitujte pomocí počítače **evil1**. Pokud výpis paměti umožňuje další způsoby kompromitace, vyzkoušejte je.

5.3.3 Opatření proti zranitelnosti

Na základě teoretických poznatků navrhněte způsob zabezpečení serveru. Cílem není praktické zabezpečení konkrétního serveru. Popište obecné postupy, které by vedly k odstranění zranitelnosti Heartbleed.

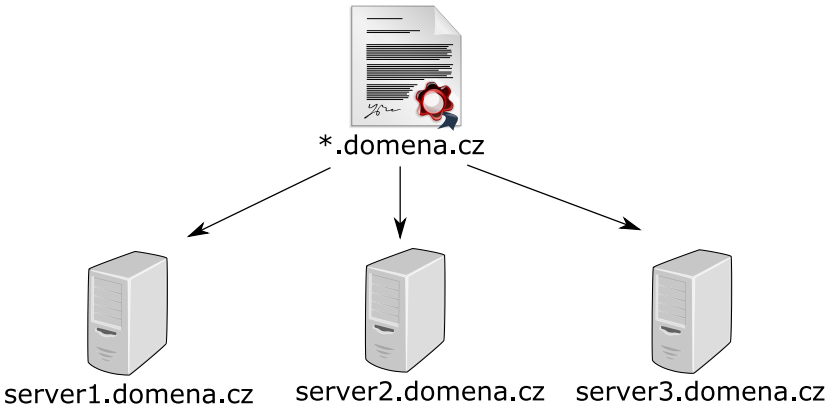
Otázka č. 6 - Navrhněte možné způsoby nápravy zranitelnosti na serveru.

6 ZNEUŽITÍ SOUKROMÉHO KLÍČE

Jméno útoku	Obtížnost	Časové nároky	Znalost	Dopad	Celkové riziko
Zneužití soukr. klíče	4	3	3	5	5

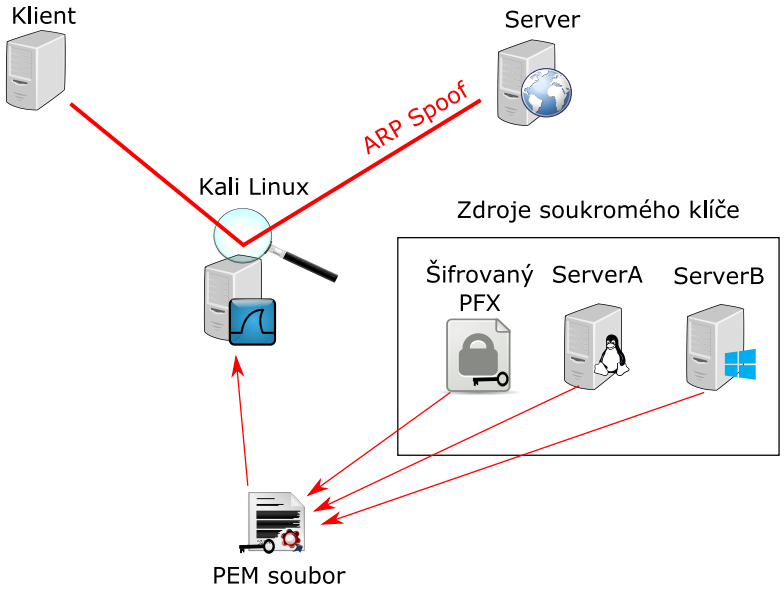
6.1 Teoretický úvod

V této kapitole bude popsána metoda útoku, která umožňuje kompromitaci SSL kanálu. Útok tohoto typu je možný pouze v případě, kdy lze získat soukromý klíč k využívanému certifikátu pro komunikaci. Na první pohled se může zdát, že je tento přístup náročný, pro interního zaměstnance ale může být získání certifikátu se soukromým klíčem snadnější než některé útoky na zranitelnosti webových serverů. Pokud útočník získá přístup k soukromému klíči, je schopný odposlouchávat většinu komunikace mezi serverem a klientem. Soukromý klíč není většinou dostupný bez další ochrany, ale její prolomení není vždy problematické. Riziko zneužití soukromého klíče roste v situacích, kdy se využívá sdíleného certifikátu pro více domén nebo tzv. wildcard certifikátu [21], tento typ certifikátu lze nazvat jako hvězdičkový. Přístup k certifikátu počtem zastoupení na více serverech roste a zdánlivě finančně výhodné řešení může vést k enormnímu riziku. Sdílení certifikátu je umožněno nahrazením jména serveru v plném doménovém jméně pomocí znaku „*“, kde tento znak nahrazuje krátké jméno serveru. Systémy podporující tento standard tak využívají stejného certifikátu, přičemž obrázek 6.1 popisuje příklad jeho využití na více systémech.



Obr. 6.1: Využití hvězdíčkového certifikátu

6.1.1 Topologie sítě laboratorní úlohy



Obr. 6.2: Zneužíti soukromého klíče

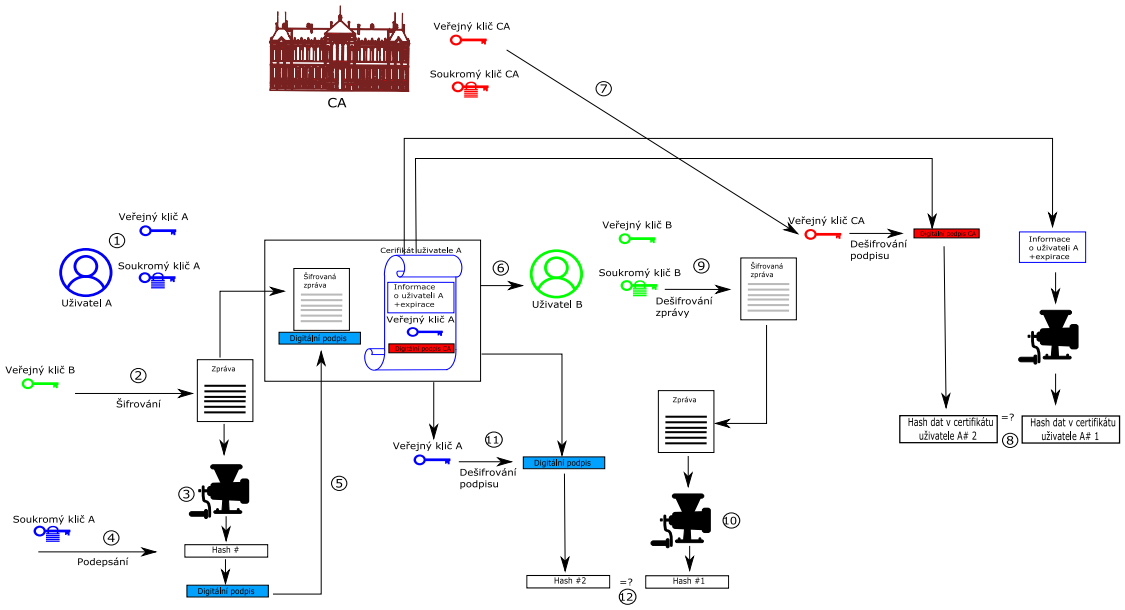
6.1.2 Certifikáty

Podstatná část zabezpečení SSL/TLS je založena na využití certifikátů. Správné pochopení funkce certifikátů je jedním ze základních stavebních kamenů dnes využívaných zabezpečení nejen u SSL/TLS.

Definice pojmu certifikát byla uvedena v RFC jako „dokument, který osvědčuje pravdivost něčeho nebo vlastnictví něčeho“ [22]. Předpokladem pro pochopení principu použití certifikátu je pochopení asymetrického šifrování a základního principu hashování. Tyto metody jsou velice dobře popsány v již zmíněné knize [15], popř. RFC [10].

Obrázek 6.3 popisuje běžné využití certifikátu a způsob, jakým protější uživatel B ověřuje uživatele A pomocí poskytnutého certifikátu uživatele A. Certifikát zde bude použit k autentizaci, zajištění integrity a zajištění důvěrnosti informací v přenášené zprávě.

1. Uživatel A má dvojici klíčů, jeden je veřejný a druhý je soukromý. Veřejný klíč byl podepsán důvěryhodnou certifikační autoritou spolu s informacemi o uživateli, čímž vznikl certifikát uživatele A.
2. Uživatel A využije veřejný klíč uživatele B pro zašifrování zprávy určené straně B. Tento veřejný klíč získal uživatel A z certifikátu uživatele B.
3. Uživatel A dodatečně vypočítá hash z posílané zprávy.
4. Tento hash je podepsán soukromým klíčem uživatele A.



Obr. 6.3: Využití certifikátu

5. Podpis je připojen k zašifrované zprávě.
6. Uživatel A posílá uživateli B takto připravenou zprávu spolu s certifikátem uživatele A.
7. Uživatel B získá certifikát certifikační autority [1]a pomocí certifikátu dešifruje digitální podpis v certifikátu uživatele A, čímž uživatel B získá předpokládaný hash certifikátu uživatele A #2. Uživatel B dále vypočítá z dat certifikátu uživatele A hash certifikátu uživatele A #1.
8. Uživatel B porovná hodnoty hash certifikátu uživatele A #1 a hash #2. Pokud jsou data identická, uživatel B může předpokládat, že nedošlo ke změně informací v certifikátu a lze ho považovat za důvěryhodný.
9. Uživatel B dešifruje svým soukromým klíčem přijatou zprávu. Výstupem je tedy dešifrovaná zpráva.
10. Uživatel B vypočítá z přijaté, dešifrované zprávy hash #1.
11. Uživatel B využije data z certifikátu uživatele A a dešifruje digitální podpis přijaté zprávy pomocí veřejného klíče uživatele A. Tím získá předpokládaný hash #2.
12. Uživatel B porovná hodnotu hashů zprávy #1 a #2. Pokud jsou identické, uživatel B může očekávat, že data jsou autentická a že zprávu odeslal uživatel A.

[1]Obě strany přitom certifikátu a certifikační autoritě věří.

6.1.3 Diffie-Hellman sestrojení klíče

Výše uvedený postup popisoval využití RSA klíčů pro vytvoření bezpečného šifrovaného kanálu. U SSL se tento kanál využívá pro přenos symetrického klíče využívaného k šifrování dat. Další způsob distribuce klíče je pomocí sestrojení klíče. V SSL lze využít k sestrojení klíče DH popsaný v knize Rolfa Oppligera [15].

Algoritmus vychází z vlastností generátoru g, pro který platí, že $g^k \mod p$ generuje všechny hodnoty z množiny hodnot ležících v intervalu celých čísel $\{1, ..., p-1\}$, taková skupina se pak označuje jako $\mathbb{Z}_p^*$. Dále je nutné nadefinovat velké prvočíslo p. Na straně A je zvolen tajný exponent $x_a \in \{0, ..., p-2\}$, následně se vypočítá veřejný exponent y_a podle předpisu:

$$y_a = g^{x_a} \mod p. \tag{6.1}$$

Podobným způsobem se postupuje na straně B, kde je zvolen tajný exponent $x_b \in \{0, ..., p-2\}$, následně se vypočítá veřejný exponent y_b podle předpisu:

$$y_b = g^{x_b} \mod p. \tag{6.2}$$

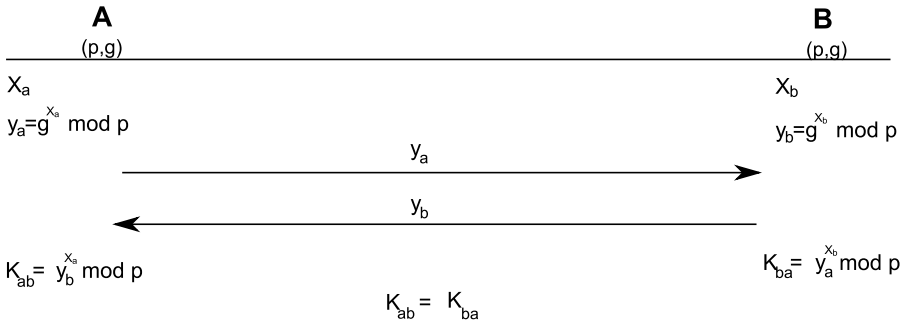
Pouze tyto veřejné exponenty jsou poslány druhé straně veřejným kanálem. Druhá strana přijme veřejný exponent druhé strany. Na straně B se vypočítá klíč pomocí tohoto předpisu:

$$K_{ba} = y_a^{x_b} = g^{x_a{}^{x_b}} \mod p = g^{x_a \cdot x_b} \mod p. \tag{6.3}$$

Na straně A se postupuje podobným způsobem a klíč se generuje pomocí tohoto předpisu:

$$K_{ab} = y_b^{x_a} = g^{x_b{}^{x_a}} \mod p = g^{x_b \cdot x_a} \mod p. \tag{6.4}$$

Vzhledem k tomu, že platí komutativní zákon exponentů $K_{ab} = K_{ba}$, jsou klíče shodné. Tyto klíče slouží k symetrickému šifrování SSL dat během komunikace. Celý proces konstrukce klíče je graficky znázorněn na obrázku 6.4.



Obr. 6.4: Sestrojení klíče DH algoritmem

6.1.4 Formáty souborů s certifikáty

Formátů souborů je velké množství, zejména proto, že způsob zpracování certifikátu na různých platformách se liší. Linuxová platforma většinou ukládá certifikáty jako oddělené soubory, které se importují do složek pro specifickou službu. U platformy Windows se importují do systému, kde jsou zabezpečeny v tzv. „certificate store“. Uživatel tak nemá možnost certifikáty přímo souborově spravovat.

PEM formát

Jedná se o nejpoužívanější formát, který využívají certifikační autority. PEM formát má většinou přípony: .pem, .crt, .cer nebo .key. Tyto soubory jsou kódovány v Base64 ASCII souborech.

Obsah souborů většinou začíná textem „-----BEGIN CERTIFICATE-----“ a je ukončen textem „-----END CERTIFICATE-----“. Soubor může obsahovat server certifikáty, certifikáty certifikačních autorit a soukromé klíče. Servery jako Apache obvykle ukládají klíče/certifikáty do oddělených souborů podle způsobu použití, přitom certifikát má obvykle příponu .crt nebo .cer a soukromý klíč má obvykle příponu .key.

DER formát

Představuje obdobu PEM formátu, jen s tím rozdílem, že se namísto ASCII formátu jedná o binární soubor. Tento typ souboru má příponu .der nebo .cer, proto může snadno dojít k záměně se souborem v PEM formátu. Rozdíl lze poznat v případě otevření souboru v textovém editoru. DER soubor neobsahuje textové návěští o začátku a konci certifikátu.

PKCS#7/P7B formát

Jedná se o soubory, které jsou uloženy v kódování Base64 ASCII. Většinou mají přípony .p7b nebo .p7c. Certifikát začíná textem „-----BEGIN PKCS7-----“ a končí textem „-----END PKCS7-----“. Tyto soubory obsahují pouze certifikáty serverů a certifikačních autorit, neobsahují soukromé klíče.

PKCS#12/PFX formát

Je binární soubor, který může obsahovat certifikáty serverů, certifikačních autorit a soukromé klíče. Tyto soubory mají obvykle příponu .pfx nebo .p12. Pfx soubory jsou obvykle využívány na platformě Windows pro import a export certifikátů a soukromých klíčů.

6.1.5 Konverze formátů certifikátů

Z PEM formátu

Konverze PEM na DER:

```
openssl x509 -outform der -in certificate.pem -out certificate.der
```

Konverze PEM na P7B:

```
openssl crl2pkcs7 -nocrl -certfile certificate.cer -out certificate.p7b -certfile CA-Cert.cer
```

Konverze PEM na PFX:

```
openssl pkcs12 -export -out certificate.pfx -inkey privateKey.key -in certificate.crt -certfile CACert.crt
```

Z DER formátu

Konverze DER na PEM:

```
openssl x509 -inform der -in certificate.cer -out certificate.pem
```

Z P7B formátu

Konverze P7B na PEM:

```
openssl pkcs7 -print_certs -in certificate.p7b -out certificate.cer
```

Z PFX formátu a opačně

Konverze PEM na PFX:

```
openssl pkcs12 -export -in certificate.pem -inkey privateKey.key -out certificate.pfx -certfile CACert.cer
```

Konverze z PFX na PEM:

```
openssl pkcs12 -in certificate.pfx -out certificate.pem -nodes
```

Konverze formátu PEM na RSA

```
openssl rsa -in certificate.pem -out priveKey.key
```

6.1.6 Způsob výměny klíčů u SSL/TLS

Před spuštěním šifrované komunikace pomocí tajného klíče dvou stran je nutné zajistit výměnu informací přes nezabezpečený kanál. Aby se zajistila důvěrnost, autenticita a integrita těchto informací, využívá se řady metod vyjednání klíčů. Tyto metody lze rozdělit do dvou základních principů. První z nich spočívá v přenesení tajného klíče jinou šifrovanou komunikací, např. pomocí asymetrického šifrování, druhý princip spočívá ve využití metody sestrojení tajného klíče. Postupně zde budou popsány jednotlivé metody výměny klíčů.

RSA

Klient generuje náhodné číslo a to zašifruje veřejným klíčem, který získá z certifikátu serveru. Přitom je nutné, aby použití certifikátu bylo určeno k šifrování. Metoda tedy neprovádí sestrojení klíče, ale veškeré informace nutné k sestavení jsou buď v nezašifrované podobě, nebo zašifrované pomocí RSA.

DH_RSA

Při této metodě dochází k sestrojení klíče z informací, které nikdy komunikačním kanálem neprošly, a z veřejných informací. V tomto případě je využíváno informací v certifikátu, kde je uvedena hodnota g^X.

Klient posílá zprávou ClientKeyExchange g^Y, přičemž tato zpráva je zašifrována veřejným klíčem z certifikátu serveru. Po výměně nezbytných informací dochází k sestrojení klíče, který je využit pro komunikaci. Podmínkou funkčnosti je RSA typ klíče v celém řetězení certifikátu. Certifikační autorita musí mít také certifikát založený na RSA klíči.

DH_DSS

Tato metoda je totožná s metodou DH_RSA, jediným rozdílem je využití DSA klíčů namísto RSA. Opět zde platí podmínka využití DSA klíče i u certifikační autority.

DHE_RSA

Princip je velice podobný jako u DH_RSA. V tomto případě ovšem není g^X součástí certifikátu serveru. Server si veřejnou část Diffie-Hellman parametru generuje dynamicky, následně ji podepsanou posílá klientovi. Klient posílá zprávu g^Y zašifrovanou zpět serveru. Certifikát serveru musí být typu RSA a použití certifikátu musí umožňovat šifrování a podepisování.

DHE_DSS

Tato metoda je totožná s metodou DH_RSA, jediným rozdílem je využití DSA klíčů namísto RSA. Opět zde platí podmínka využí DSA klíče u serveru.

DH_ANON

V tomto případě se nevyužívá certifikátu serveru ani pro šifrování, ani pro podepisování. Veřejné parametry Diffie-Hellman algoritmu se posílají bez dalšího šifrování v obou směrech. Metoda je náchylná na MITM útok, proto by se neměla používat.

ECDH_RSA

Pří výpočtu se využívá šifrování pomocí eliptických křivek. Samotný princip zpracování veřejných parametrů je podobný jako u DH_RSA. Všechny zřetězené certifikáty musí využívat RSA certifikáty s pevně danými parametry eliptických křivek v certifikátech.

ECDH_ECDSA

Princip i pravidla pro certifikáty jsou shodné s ECDH_RSA. Rozdíl je v tom, že u této metody se využívá DSA certifikátů.

ECDHE_RSA

Server vygeneruje dynamicky parametry pro eliptické šifrování a podepisuje parametry serverovým certifikátem, který musí být typu RSA. Princip výměny klíčů je podobný DHE_RSA.

ECDHE_ECDSA

Server vygeneruje dynamicky parametry pro eliptické šifrování a podepisuje parametry serverovým certifikátem, který musí být typu ECDSA. Princip výměny klíčů je podobný DHE_DSS.

6.1.7 Využití programu Wireshark

Wireshark je velice efektivní nástroj pro analýzu síťového provozu v řadě situací. V tomto případě lze využít možností přímého dešifrování HTTPS provozu, který lze odposlechnout ze síťového provozu. K odposlechnutí provozu je nutné získat tajný klíč, který se využívá k šifrování dat.

K získání tajného klíče je nutné prolomit výměnu tajného klíče, která je zabezpečena různými způsoby. Jednou z variant je zabezpečení pomocí asymetrického šifrování prostřednictvím RSA klíčů. Pokud útočník získá přístup k soukromému klíči RSA, lze jej naimportovat do programu **Wireshark**. **Wireshark** následně provede výpočet tajného klíče z PMS a dalších proměnných a přímo dešifruje komunikaci. Pro import soukromého klíče je nutné převést klíč do formátu PEM, tento soubor pak lze naimportovat do programu.

6.2 Praktická část

Tab. 6.1: Seznam počítačů a IP adresy

Jméno počítače	IP adresa
cl1	192.168.64.137
evil1	192.168.64.136
srv1	192.168.64.135
srv2	-
srv3	-

V praktické části bude realizován komplexní útok na sdílený hvězdičkový certifikát použitý v testovacím prostředí. Certifikát může být v praxi použit na více systémech a má stejnou soukromou část. Kompromitace soukromého klíče tak umožňuje kompromitovat komunikaci jakéhokoliv serveru, který využívá sdíleného certifikátu při komunikaci. Správa soukromého klíče certifikátu bývá alespoň minimálně zabezpečena, nedůsledné zabezpečení ovšem umožňuje získat soukromou část relativně snadno.

V praktické části bude realizováno několik útoků, které povedou k získání soukromé části klíče. Výsledek všech útoků je stejný, ale kombinací možných útoků útočník zvýší pravděpodobnost získání soukromé části certifikátu. **Získejte soukromý klíč alespoň prolomením pfx souboru slovníkovým útokem a z jednoho fyzického útoku na server dle Vašeho uvážení.**

Po získání soukromé části certifikátu bude analyzován provoz pomocí programu **Wireshark** v kombinaci s **ARP spoof**.

6.2.1 Získání soukromého klíče z pfx souboru

Certifikáty se obvykle ukládají ve formátu, který je zašifrovaný. Takto zabezpečený soubor lze snadněji bezpečně předávat přes méně bezpečný kanál. Šifrování pfx souboru může být realizováno pomocí hesla. Pokud je heslo nevhodně zvoleno, umožňuje

útočníkovi využití slovníkového útoku, popř. útoku pomocí postupného generování hesel, tzv. „brute force“ útoku. Obvykle se spoléhá na zabezpečení samotného pfx souboru, proto je přístup k souboru jen minimálně chráněn. Pomocí dalších metod lze získat požadované oprávnění a získat tak přístup k pfx souboru. V laboratorní úloze jste v situaci, kdy máte pfx soubor, ale neznáte heslo k rozšifrování souboru.

K prolomení hesla pfx souborů lze využít řady volně dostupných programů. V této úloze bude použit program **Crackpkcs12**, který je volně dostupný [1]. Program umožňuje více postupů k prolomení hesla. V této laboratorní úloze využijete tzv. slovníkový útok. Slovník je v tomto případě textový soubor obsahující nejčastěji používaná hesla. Využitím slovníku předejdete útoku pomocí náhodně generovaných hesel. Využití slovníku však omezuje seznam hesel a je možné, že heslo nebude ve slovníku obsaženo.

Program **Crackpkcs12** je již nainstalován na počítači **evil1**. Všechny soubory nutné k prolomení pfx souboru jsou uloženy ve složce **Desktop**. Zde také naleznete slovník s hesly **passwords.txt** a pfx soubor **wildcard.pfx**, který obsahuje certifikát včetně soukromého klíče, chráněný heslem. Počítač **evil1** obsahuje dva procesory, program lze tedy spustit ve dvou vláknech a tak urychlit prolomení hesla.

Spusťte počítač **evil1**, seznamte se s programem **Crackpkcs12** a proveďte slovníkový útok pomocí slovníku **passwords.txt** na pfx soubor **wildcard.pfx**.

Prolomení hesla může trvat přibližně 15 minut na systému s jedním procesorem, namísto čekání na dokončení běhu programu lze pracovat na dalších částech laboratorní úlohy.

Při reálném testování hesel je obvykle nutné určit očekávanou dobu prolomení hesla. Program **Crackpkcs12** informuje o statistikách počtu otestovaných hesel během jeho spuštění. Z informací o počtu testovaných hesel/řádků za sekundu lze při zjištění počtu řádků ve slovníku vypočítat maximální přibližnou dobu útoku. [2]

Otázka č. 1 - Z informací o rychlosti programu viditelných při běhu určete maximální možnou dobu útoku při použití dat ze slovníku **passwords.txt**. Přibližnou hodnotu uveďte v protokolu.

[2] Pro výpočet počtu řádků lze v Linuxu využít program **Word Count** s názvem **wc**. Pro zvolení správných parametrů využijte manuál k programu.

Otázka č. 2 - Zjistěte, jakým heslem je chráněn pfx soubor. Heslo následně uveďte v protokolu.

K využití soukromého klíče je nutné klíč z pfx souboru vyexportovat. Pro vyexportování klíče použijte získané heslo a převeďte klíč na RSA formát, který je nutný pro použití v programu **Wireshark**.

Na základě informací v teoretické části vyexportujte soukromý klíč v RSA formátu.

6.2.2 Získání soukromého klíče z Windows serveru

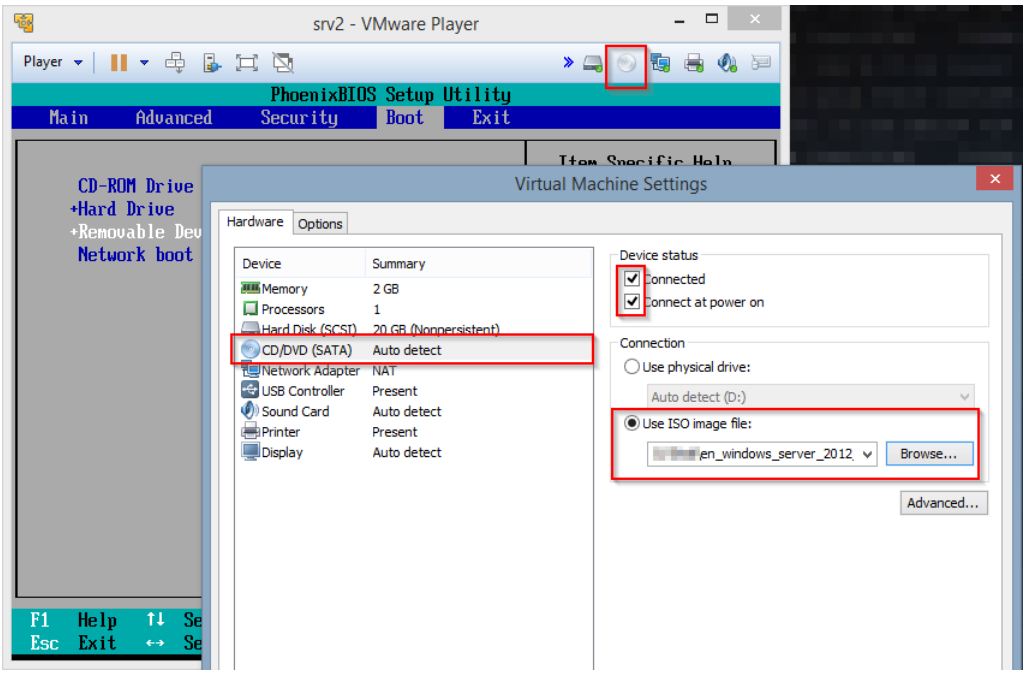
Windows systém spravuje certifikáty v tzv. „certificate store“, ke kterému je omezený přístup. V případě, že certifikát nebyl označen jako exportovatelný, nelze soukromou část certifikátu již získat. V mnoha případech však je certifikát exportovatelný, i když k tomu nemusí být důvod. V této části laboratorní úlohy získáte fyzický přístup k serveru s operačním systémem Windows 2012 R2. Tento server byl vyřazen z provozu a byl uložen ve skladu, kde systém čeká na smazání dat z disků a na následnou fyzickou likvidaci serveru. Přístup k serveru ovšem není natolik chráněn a k serveru mají přístup i osoby, které mohou využít fyzického útoku na stroj.

Vy, jako útočník, máte k dispozici instalační médium Windows 2012 R2, které lze pořídit bezplatně na internetu, máte k serveru fyzický přístup a systém obsahuje optickou mechaniku. Pokud není disk šifrovaný, fyzický přístup je obecně považován za plný přístup k systému. Existuje velké množství způsobů jak systém kompromitovat. V této úloze byl zvolen záměrně způsob, který nevyžaduje žádné specifické nástroje, což zobecňuje využití této praktiky.

Spusťte počítač **srv2**, upravte konfiguraci priorit bootování v BIOSu a připojte obraz instalačního média.Systém nabootujte z obrazu.

K útoku bude využit počítač **srv2** [3]. Ten má nainstalovaný operační systém Windows 2012 R2 a obsahuje hvězdičkový certifikát i se soukromou částí klíče. Po spuštění systému mačkejte klávesu **F2** tak, aby počítač otevřel nastavení BIOSu. V záložce **Boot** změňte pořadí bootu ze zařízení tak, že první zařízení bude **CD-ROM Drive**, druhé zařízení by mělo být **Hard Drive**. Dál s nastavením BIOSu

[3] Během útoku server nevypínejte, používejte pouze restart. Virtuální stroj je nepersistentní, vypnutí systému smaže veškeré změny a tím i Vaši práci na systému.

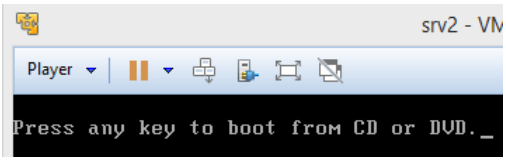


Obr. 6.5: Připojení obrazu

nic nedělejte a uvolněte ovládání virtuálního stroje. Bude nutné nejprve změnit nastavení v konfiguraci programu **VMWare Player**.

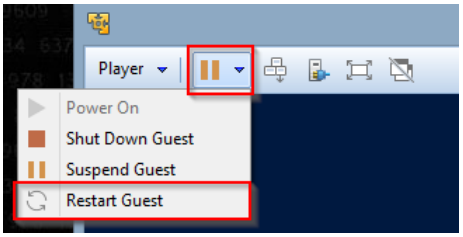
V tuto chvíli bude nutné připojit obraz instalačního média Windows 2012 R2. Instalační médium je uloženo ve stejném umístění, kde se nacházejí virtuální stroje. Obraz připojíte klepnutím pravého tlačítka na ikonu CD v menu otevřeného programu **VMWare Player**, následně klepněte na volbu **Settings...**. Po otevření nastavení vyberte v levé části zařízení optické médium, v pravé části ověřte, že nastavení **Device status** odpovídá nastavení na obrázku 6.5.

Vyberte obraz instalačního média, jméno souboru je: **en_windows_server_2012_r2_x64_dvd_2707946.iso**. Potvrďte nastavení tlačítkem **OK**. Po tomto nastavení se připojte zpět do BIOSu stroje a stiskněte klávesu **F10**. Potvrďte uložení změn klávesou **Y**.Virtuální počítač se restartuje a začne bootovat z optického média. Tento boot je ale nutné potvrdit stisknutím jakékoliv klávesy, viz obrázek 6.6. V opačném případě systém nabootuje z disku.



Obr. 6.6: Potvrzení bootování z CD/DVD

Pokud včas nezareagujete a nestisknete libovolnou klávesu, proveďte restart systému pomocí ovládání virtuálního stroje, viz obrázek 6.7. **V žádném případě systém nevypínejte, pouze systém restartujte.**



Obr. 6.7: Restart systému

Na počítači **srv2** zvolte opravu počítače a spusťte příkazovou řádku.

Po nabootování počítače pomocí připojeného obrazu se spustí instalační program Windows 2012 R2. V prvním okně ponechejte vše beze změny a stiskněte tlačítko **Next**. V dalším okně je možná volba **Repair your computer**, stiskněte tuto volbu. V okně **Choose an option** vyberte položku **Troubleshoot**. Ve výběru **Advanced options** zvolte položku **Command Prompt**. Tím spustíte příkazovou řádku, pomocí které budete modifikovat nainstalovaný systém.

Na počítači **srv2** spusťte tyto příkazy:

- **C:**
- **cd Windows\System32**
- **ren Utilman.exe Utilman.exe.bak**
- **copy cmd.exe Utilman.exe**.

Výše uvedené příkazy přejmenují program **Utilman.exe** a namísto něj vytvoří kopii programu **Cmd.exe** přejmenovanou na **Utilman.exe**.

Otázka č. 3 - Identifikujte, k čemu se využívá ve Windows program **Utilman.exe** a k čemu slouží program **Cmd.exe**. Pokuste se odhadnout, proč byl tento krok nutný a co způsobí.

Restartujte počítač **srv2** pomocí příkazové řádky.

Ve volbách instalace je pouze vypnutí systému, proto bude nutné vynutit restart systému z příkazové řádky. Pro restart systému využijte program **shutdown.exe** s parametry:

- **-r**
- **-t 0**

Parametr **-r** určuje restart namísto vypnutí systému. Parametr **-t** určuje, za jak dlouhý časový okamžik má dojít k restartu, popř. vypnutí. Následuje číselná hodnota určující počet sekund do restartu, popř. vypnutí.

> Klikněte na **usnadnění přístupu** v levém spodním rohu. V příkazové řádce pak změňte heslo účtu **admin** na heslo **Heslo123**. Následně se pomocí účtu admin přihlaste do systému.

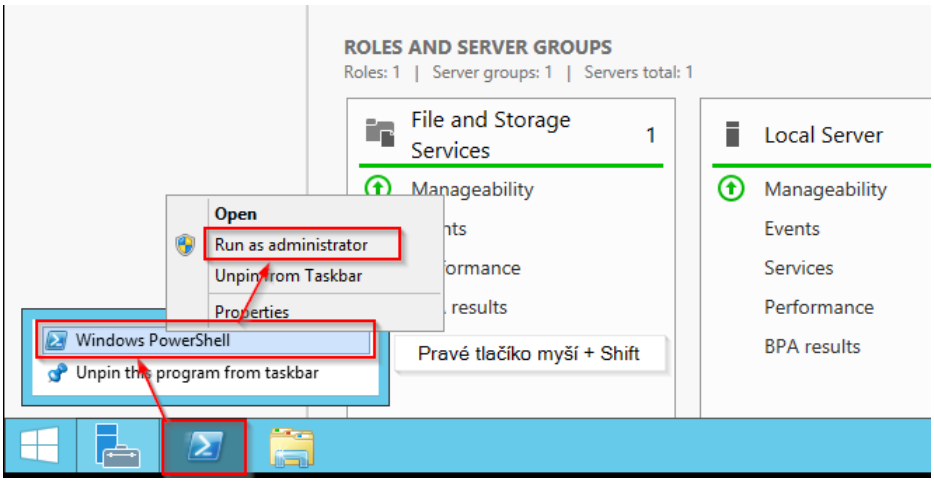
Po spuštění systému stiskněte tlačítko v levém spodním rohu okna, jedná se o program pro usnadnění přístupu. Po stisknutí tlačítka však namísto usnadnění přístupu dojde ke spuštění příkazové řádky. V příkazové řádce změňte heslo účtu **admin**. To provedete pomocí příkazu:
net user admin Heslo123. Následně můžete příkazovou řádku zavřít a pomocí právě nastaveného hesla se k systému přihlaste.

> Na počítači **srv2** spusťte správu certifikátu v privilegovaném režimu. Následně vyexportujte certifikát včetně soukromého klíče.

Po přihlášení do systému bude nutné spustit správu certifikátů serveru. Postupů pro její spuštění je více a mohou se lišit podle verze operačního systému. Pravým tlačítkem klikněte na ikonu **Powershell**. Následně přidržte levé tlačítko **Shift** a pravým tlačítkem klikněte na položku **Windows PowerShell**. V nově otevřeném okně vyberte **Run as administrator**, viz obrázek 6.8.

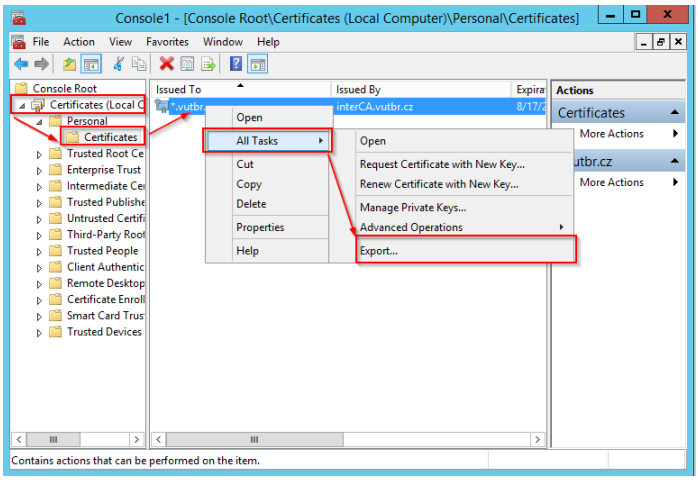
Takto spustíte příkazovou řádku s oprávněním administrátora. V nově otevřeném programu **Powershell** spusťte příkaz **certlm.msc**, tím otevřete správu certifikátů. Ve správci pak spusťte export certifikátu, viz obrázek 6.9.

V novém okně pro export certifikátu potvrďte export tlačítkem **Next**. V následujícím okně lze zvolit, jestli má být vyexportován pouze certifikát, nebo má dojít také k exportu soukromého klíče. Zde zvolte **Yes, export the private key** a volbu potvrďte tlačítkem **Next**. V okně **Export File Format** ověřte volbu exportu do



Obr. 6.8: Spouštění přivilegovaného interpretu

formátu PFX, dále ověřte zaškrtnutí volby **Include all certificates in the certification path if possible** a volbu potvrďte tlačítkem **Next**. V okně **Security** zvolte heslo, kterým bude pfx soubor zabezpečen. Heslo si zapamatujte, bude potřeba při exportu soukromého klíče z tohoto souboru. V okně **File to Export** zvolte jméno pfx souboru, volbu potvrďte tlačítkem **Next**. V následujícím okně ověřte volby a stiskněte tlačítko **Finish**.



Obr. 6.9: Export certifikátu

Soubor vyexportujte přetáhnutím z okna virtuálního stroje na laboratorní počítač. Následně tento pfx soubor podobným způsobem naimportujte na počítač **evil1** a proveďte export soukromého klíče ve formátu RSA.

Naimportujte do počítače **evil1** pfx soubor vyexportovaný z počítače **srv2** a vyexportujte z tohoto pfx souboru soukromý klíč ve formátu RSA.

Otázka č. 4 - Porovnejte obsah soukromého klíče v RSA formátu. Je tento soukromý klíč identický s klíčem získaným v první části, kde bylo prolomeno heslo k pfx klíči?

6.2.3 Získání soukromého klíče z Linux serveru

V předchozí části jste získali soukromý klíč z Windows serveru pomocí fyzického útoku na již nepoužívaný server. Situace může být velmi podobná i v případě, že server obsahuje jiný operační systém. V tomto případě půjde o server s operačním systémem Linux. Cílem útoku bude počítač **srv3**, který využívá hvězdičkový certifikát pro běh HTTPS serveru. Postup změny boot priorit a připojení obrazu je identický útoku na Windows server. V tomto případě ovšem bude připojen obraz **kali-linux-1.1.0-amd64.iso**.

Nastavte boot priority na počítači **srv3** tak, aby systém nejprve bootoval z optického média. Připojte obraz optického média distribuce Kali a nabootujte z něj.

Při bootu z instalačního média distribuce Kali je zobrazeno menu. Vyberte start live verze distribuce Kali, tedy první volbu **Live (amd64)**. Po nastartování dojde k automatickému přihlášení k distribuci.

Otevřete okno terminálu, bude nutné připojit disk k serveru, který obsahuje nainstalovanou distribuci. Nejprve je nutné nalézt disk a oddíly. Pomocí příkazu **lsblk** vypíšete konfiguraci disků a oddílů. Identifikujte oddíl (TYPE=part), který má velikost větší než několik GB. Jméno systémového oddílu bude začínat znaky „sd“. Jméno oddílu si poznamenejte.

Připojte systémový disk serveru ke složce **/media/** v live distribuci Kali.

Systémový disk je nutné připojit k live distribuci. To je možné realizovat pomocí programu **Mount** a disk připojit do složky **/media/**, což lze provést následovně:

mount /dev/*sdXN* /media/

sdXN reprezentuje jméno disku, který jste identifikovali jako systémový disk, přičemž symbol ***X*** představuje znak a symbol ***N*** představuje číselnou hodnotu.

Po spuštění příkazu **mount** lze ověřit úspěšné připojení několika způsoby. Pomocí příkazu **mount | grep /media** lze zobrazit připojení disku do složky **/media/**. Dále je možné zobrazit souborový systém připojeného disku, což lze provést pomocí příkazu **ls /media/**.

Změňte root adresář live distribuce Kali na adresář /media/.

Proto, aby bylo možné modifikovat systémové informace na systémovém disku serveru, je nutné provést operaci, která změní root adresář systému live distribuce Kali na adresář **/media/**, tedy na systémový disk serveru. Změnu root adresáře lze provést pomocí **chroot**. Pro změnu adresáře tedy použijte příkaz:

sudo chroot /media/

V tuto chvíli máte plný přístup k systému a veškeré akce provedené v otevřeném terminálu budou provedeny na systémovém disku serveru. Těchto oprávnění využijete pro zjištění informací o uživatelských účtech a ke změně hesla účtu, který má přístup jako administrátor.

Identifikujte privilegované účty a změňte heslo privilegovaného účtu.

Pro zjištění uživatelských účtů, které mají nastavené heslo, lze využít příkazu **cat**, pomocí kterého lze vypsat obsah souboru **shadow**. Tento soubor obsahuje záznamy o hash záznamech hesel, není proto možné hesla vyčíst přímo. Tímto způsobem lze ale zjistit, u kterých účtů je heslo nastaveno, a může se jednat o privilegovaný účet.

cat /etc/shadow | grep "\$1"

Pokud se jedná o distribuci, která využívá příkaz **sudo**, lze zjistit privilegované uživatele a skupiny pomocí vyčtení informací ze souboru **sudoers**. V tomto souboru je zapsaná skupina, která umožňuje uživatelům provádět oprávnění s právy uživatele **root**. Na tomto konkrétním systému je privilegovaná skupina **admin**.

```
cat /etc/sudoers
```

Pro zjištění členů skupiny **admin** lze využít obsahu souboru **group**. Tím identifikujete uživatele, kterému změníte heslo. Jedná se o uživatele **bee**.

```
cat /etc/group | grep "admin"
```

Změňte heslo privilegovaného uživatele v systému.

Pro změnu uživatelského účtu lze využít programu **Passwd**, jako parametr zadejte jméno privilegovaného účtu. Následně budete vyzváni k zadání hesla. Reboot systému můžete spustit pomocí příkazu **sudo reboot** v novém terminálovém okně. Po rebootu bude systém opět bootovat z optického média, proto bude nutné buď změnit boot sekvenci, nebo odpojit optické médium v nastavení. **V žádném případě nevypínejte systém. Pokud je nutné, provádějte v ovládání virtuálního počítače pouze reboot, jinak systém ztratí provedené změny!** Po rebootu systému budete vyzváni k zadání uživatelského jména a hesla. Vzhledem k nestandardnímu rozložení klávesnice je doporučený postup zadání hesla do okna, do kterého se zadává uživatelský účet. Následně vyberte text hesla a vložte jej do schránky pomocí **Ctrl + x**. Zadejte uživatelský účet a při výzvě k zadání hesla využijte heslo uložené ve schránce a toto heslo vložte pomocí **Ctrl + v**.

Korektně restartujte systém a odpojte obraz optického média. Přihlaste se k privilegovanému uživateli pomocí nového hesla. Pozor na rozložení klávesnice!

V linuxových systémech se obvykle využívá přímo čitelný formát soukromých klíčů. Složky jsou chráněny pouze přístupovými právy. Vzhledem k tom, že byl v našem případě kompromitován účet s právy **sudoers**, lze soukromý klíč získat. Cesta k souboru k certifikátu se soukromým klíčem je **/etc/ssl/certs/bee-box.pem**.

```
sudo cp /etc/ssl/certs/bee-box.pem ./Desktop/
sudo chmod +rw ./Desktop/bee-box.pem
```

Soubor se soukromým klíčem vykopírujte do složky Desktop, změňte práva k souboru a vyexportujte soubor z virtuálního stroje. Vyexportovaný soubor upravte tak, aby obsahoval pouze soukromý klíč, a uložte soubor s příponou .key.

Otázka č. 5 - Porovnejte obsah výsledného souboru se soukromými klíči z předchozích dvou způsobů získání soukromého klíče. V protokolu porovnejte metody podle technické a časové náročnosti.

6.2.4 Zachycení a dešifrování komunikace

V předchozích třech částech byly realizovány způsoby, které vedly k získání soukromého klíče k hvězdičkovému certifikátu využívaném i na cíli **srv1**. V této části bude využit soukromý klíč k dešifrování komunikace mezi počítačem **cl1** a počítačem **srv1**.

K dešifrování komunikace je nutné provést několik důležitých kroků. Nejprve je nutné naimportovat na počítači **evil1** soukromý klíč do programu **Wireshark**. Následně bude nutné spustit na počítači **evil1** přesměrování příchozího provozu pomocí pravidla v **IPtables** a spustit útok ARP spoof, který zajistí vynucení přesměrování provozu mezi počítačem **cl1** a počítačem **srv1**.

Naimportujte upravený soubor se soukromým klíčem na počítač **evil1**, spusťte program **Wireshark** a naimportujte do něj soukromý klíč. Spusťte počítače **cl1** a **srv1**.

Přesunutím souboru na okno virtuálního stroje **evil1** naimportujete na plochu soubor se soukromým klíčem. Program **Wireshark** spustíte buď z terminálu, nebo kliknutím na **Applications -> Internet -> Wireshark**. Pro import soukromého klíče do programu **Wireshark** klikněte na **Edit -> Preferences**. V okně **Preferences** vyberte v levé části okna **Protocols -> SSL**, v pravé části okna pak klikněte na tlačítko **Edit** u volby **RSA keys list**. V nově otevřeném okně pak klikněte na tlačíko **New**. Tímto otevřete nové okno, kde se konfigurují podmínky pro dešifrování provozu. Do volby **IP address** zadejte IP adresu počítače **srv1**. Do volby **Port** zadejte **443**, do volby **Protocol** zadejte **http**. Ve volbě **Key File** klikněte na tlačítko složky a vyberte soubor se soukromým klíčem ve formátu RSA. Následně potvrďte výběr souboru tlačítkem **Open**. Pro dokončení konfigurace pravidel

pro dešifrování stiskněte tlačítko **OK**. Následně potvrďte změnu okna **Preferences** kliknutím na tlačítko **OK**.

Na počítači **evil1** spusťte přesměrování příchozího provozu a spusťte ARP spoof.

Nejprve je nutné spustit přesměrování IP provozu, nastavení NAT [25] a úpravy pravidel firewallu.

```
echo "1" > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables -A FORWARD -i eth0 -j ACCEPT
```

Následně spusťte MITM útok pomocí ARP spoof, v samostatných oknech terminálu spusťte příkazy:

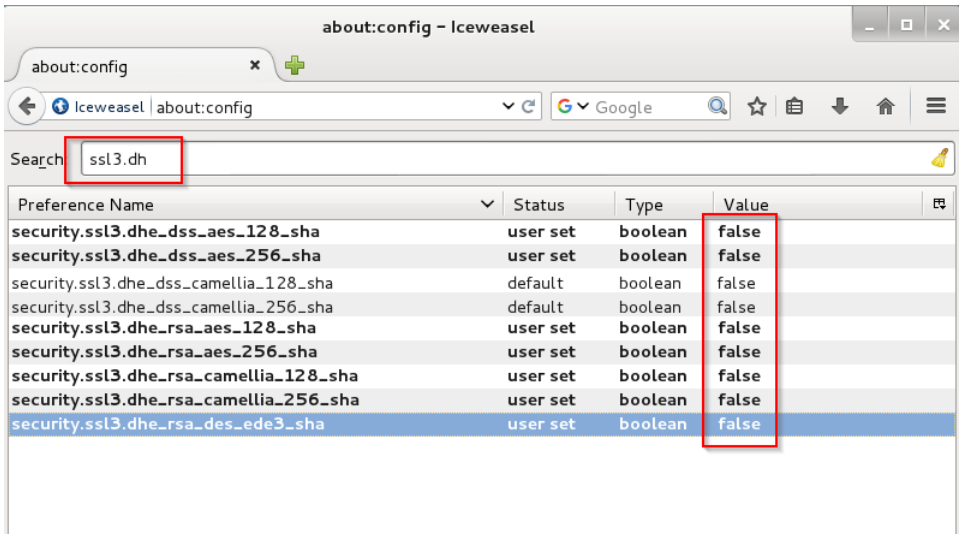
```
arpspoof -i eth0 -t 192.168.64.137 192.168.64.135
arpspoof -i eth0 -t 192.168.64.135 192.168.64.137
```

Na počítači evil1 spusťte odposlech v programu Wireshark provozu na rozhraní eth0.

Pro spuštění odposlechu na počítači **evil1** klikněte na **Capture -> Interfaces...**, v nově otevřeném okně zaškrtněte rozhraní **eth0** a stiskněte tlačítko **Start**.

Na počítači **cl1** upravte konfiguraci **Iceweasel** tak, aby nebyly použity cipher suite typu Diffie Hellman, program po změně uzavřete.

Pro zajištění úspěšného dešifrování je nutné zajistit využití cipher suite RSA, všechny typy Diffie Hellman nejsou vhodné. V programu **Iceweasel** zadejte do adresní řádky **about:config** a potvrďte varování. Následně do vyhledávacího pole zadejte „ssl3.dh“, čímž aktivujete filtraci pouze specifických nastavení. U všech zobrazených nastavení změňte hodnotu **Value = false**, viz obrázek 6.10. Program **Iceweasel** uzavřete.



Obr. 6.10: Upravení prohlížeče na klientském počítači

Na počítači **cl1** spusťte program **Iceweasel** a přihlaste se na aplikaci **bWAPP** pomocí uživatelského účtu **bee**.

Na počítači **cl1** otevřete v programu **Iceweasel** adresu **https://srv1.vutbr.cz**, klikněte na aplikaci **bWAPP** a přihlaste se pomocí účtu **bee** a hesla **bug**.

Na počítači **evil1** zastavte odposlech provozu a zachycená data analyzujte. Najděte uživatelské jméno a heslo použité během přihlášení a z příslušných zpráv pořiďte snímky, ty vložte do protokolu.

V programu **Wireshark** zastavte odposlech červeným tlačítkem v hlavním menu. Do pole **Filter** zadejte „http“. V záznamech zpráv hledejte zprávu s metodou **POST**. Ze zprávy s příslušnými přihlašovacími údaji vytvořte snímek a ten uložte do protokolu.

Otázka č. 6 - Vysvětlete, proč bylo nutné vypnout cipher suite využívající metody sestrojení klíče Diffie-Hellman.

Otázka č. 7 - Pokuste se zachytit již probíhající komunikaci bez toho, aniž by byl odposlechnut začátek této komunikace. Proč nelze komunikaci v tomto případě dešifrovat?

6.2.5 Opatření proti zranitelnosti

V této laboratorní úloze byl demonstrován útok, který souvisí se správou klíčů. Během práce na této laboratorní úloze musely být zajištěny určité podmínky, které umožnily výsledné dešifrování komunikace. Zamyslete se nad podmínkami a na základě těchto informací navrhněte vhodná opatření proti zranitelnosti.

Otázka č. 8 - Jakým způsobem by bylo možné předejít zneužití pfx souboru? Navrhněte postupy, které by omezily nebo ztížily tento útok.

Otázka č. 9 - Jakým způsobem by bylo možné předejít zneužití již nepoužívaných serverů? Navrhněte postupy, které by omezily nebo ztížily tento útok.

7 SHRNUTÍ LABORATORNÍCH ÚLOH

Důvodem pro vypracovaní této kapitoly byly dodatečné poznatky získané při realizaci laboratorních úloh a další praktické poznámky důležité během této realizace. Podobné informace by nebyly v závěru vhodné, přitom by však mohly významně usnadnit budoucí práci s laboratorními úlohami.

7.1 Práce s virtuálními stroji

7.1.1 Síťová konfigurace

Všechny virtuální stroje mají nastavenou statickou konfiguraci IP adres a konfiguraci využívající NAT rozhraní. Vzhledem k dynamicky měnícímu se IP rozsahu na počítačích s VMware nebude připojení k internetu funkční. Pokud bude nutné nastavit připojení k internetu, je nutné také nakonfigurovat danou síť následujícím způsobem:

- Rozsah sítě: **192.168.64.0/24**
- Výchozí brána: **192.168.64.2**
- DNS server: **192.168.64.2**

Během práce s virtuálními stroji bylo identifikováno nevhodné chování systémů vycházejících z distribuce Debian. V případě kopírování virtuálního systému na laboratorní počítač dejte pozor při prvním startování systému. Systém kontroluje případné manipulace strojů a při prvním spuštění po kopírování či přesunu se VMware Player dotazuje, zda byl virtuální stroj přesunut či překopírován. V případě tohoto dotazu stiskněte tlačítko **I moved it**. V případě, že potvrdíte zkopírování virtuálního stroje, dochází ke generování nové MAC adresy virtuálního stroje. Generování nové fyzické adresy může způsobit změnu pořadí rozhraní síťového pojmenování ve stroji. Např. adresa eth0 může být nahrazena adresou eth1. Takové změny způsobí vytvoření nového rozhraní a dynamického alokování IP adresy z DHCP namísto využití očekávané IP adresy.

7.1.2 Optimalizace výkonu

Při práci s virtuálními stroji byl zaznamenán výrazný výkonnostní problém při práci více systémů ze stejného fyzického disku založeného na konvenční technologii magnetického záznamu. Při monitorování systému disk vykazoval 100% zatížení s dobou odezvy až několik sekund. V takové situaci nebylo možné s virtuálními systémy pracovat v reálném čase a bylo nutné systémy vypnout a postupně spouštět

znovu. Ve chvíli, kdy bylo nutné provádět více náročné diskové operace, bylo nutné ostatní systémy vypnout. Tento problém způsobilo pravděpodobně zaplnění fronty vstupně-výstupních operací. Zaplnění této fronty zamezilo zpracování dalších požadavků a všechny virtuální systémy nebyly schopny další interakce s uživatelem. Problém však nebyl identifikován na laboratorním stroji, pravděpodobně byl disk na laboratorním stroji výkonnější. Pro měření vstupně-výstupní operace byl použit volně dostupný program Iometer [23]. Problém byl vyřešen přesunutím virtuálních strojů na SSD, přičemž požadavky byly po přesunu zpracovávány se zpožděním do 20 ms. Při měření vstupně-výstupních operací za sekundu byl konvenční disk schopen zpracovat 97 operací, oproti tomu SSD disk byl schopen zpracovat 7167 vstupně-výstupních operací za sekundu.

Při konfiguraci pracovní stanice v laboratořích lze doporučit program Iometer pro základní stanovení výkonnosti systému. Na základě zkušeností při konfiguraci virtuálních systémů je důležité zvolit výkonný disk založený na technologii SSD. Disk má zásadnější význam než výkon procesoru nebo množství paměti.

7.1.3 Připojení obrazu optických médií

U počítačů **srv2** a **srv3** je nutné připojit obrazy optických médií. Připojením optického média před prvním využitím virtuálního stroje ulehčí studentovi hledání obrazů na disku. Cesty k obrazu zůstávají v konfiguraci i poté, co byly disky odpojeny.

Dále byly oba stroje nakonfigurovány tak, aby při startu systémy čekaly 30 sekund před začátkem bootu systémů. Tato konfigurace byla realizována přidáním tohoto řádku do konfigurace .vmx souboru každého z těchto strojů:
bios.bootdelay = 30000.

7.2 Generování certifikátů

Během konfigurace laboratorních úloh bylo využito certifikační autority, která byla využita pro generování certifikátu v laboratorních úlohách. Celá infrastruktura byla původně vytvořena na počítači **evil1**, kde byla za účelem generování certifikátů uložena ve složce **/root/ca**. V příloze je k dispozici archiv **ca.zip** obsahující celou tuto infrastrukturu.

7.2.1 Vygenerování nového certifikátu

Po rozbalení celého archivu do složky **/root/ca** lze provést vygenerování nového certifikátu. Samotný soukromý klíč CA je chráněn heslem **password**. Pro generování certifikátu zadejte tyto příkazy: [1]

```
cd /root/ca
openssl genrsa -out intermediate/private/newserver.vutbr.cz.key.pem 2048
chmod 400 intermediate/private/newserver.vutbr.cz.key.pem

openssl req -config intermediate/openssl.cnf \
-key intermediate/private/newserver.vutbr.cz.key.pem \
-new -sha256 -out intermediate/csr/newserver.vutbr.cz.csr.pem
```

V tuto chvíli budete vyzváni k potvrzení parametrů certifikátu. Doporučené nastavení lze nastavit pouhým stisknutím tlačíka **Enter** pro tyto parametry:

- Country Name (2 letter code) [CZ]:
- State or Province Name [Czech]:
- Locality Name [Brno]:
- Organization Name [VUT, Brno]:
- Organizational Unit Name [FEKT]:

Následně budete vyzváni pro zadání **Common Name**. Tento parametr by měl odpovídat jménu serveru, v tomto případě tedy **newserver.vutbr.cz**. Poté budete vyzváni k zadání emailové adresy. Tuto volbu můžete ponechat ve výchozím nastavení, postačuje tedy potvrdit tlačítkem **Enter**.

- Common Name []:**newserver.vutbr.cz**
- Email Address [interadmin@vutbr.cz]:

K podepsání certifikátu certifikační autoritou použijte následující příkazy:

```
openssl ca -config intermediate/openssl.cnf \
-extensions server_cert -days 2000 -notext -md sha256 \
-in intermediate/csr/newserver.vutbr.cz.csr.pem \
-out intermediate/certs/newserver.vutbr.cz.cert.pem
```

Během procesu podepisování budete vyzváni k zadání hesla, použijte tedy heslo **password**. Následně bude nutné spustit podepisování a potvrdit podpis certifikátem CA. Pro každý tento dotaz je nutné zadat **y** a potvrdit volbu tlačítkem **Enter**.

[1] V tomto příkladě jde o generování certifikátu pro server **newserver.vutbr.cz.**

Pro dodatečné nastavení práv k nově vygenerovanému certifikátu spusťte tento příkaz:

```
chmod 444 intermediate/certs/newserver.vutbr.cz.cert.pem
```

7.3 Příručky pro učitele

V přílohách práce jsou k dipozici návody, které mají usnadnit orientaci v problematice a poskytnout tak učiteli informace, které jsou nutné pro asistenci studentům během práce na laboratorních úlohách. V přílohách jsou rovněž obsaženy odpovědi na otázky položené studentů během praktický částí. Příručky pro učitele by měly poskytnout učiteli v laboratořích dostatečné zázemí pro prověření znalostí studentů.

Příručky nejsou součástí tištěné verze práce, jsou ale k dispozici na přiloženém médiu této práce a v přílohách.

7.4 Návody pro laboratorní úlohy

V přílohách této práce jsou také návody upravené ve formátu docx. Důvodem vypracování těchto návodů a jejich zahrnutí do této práce je snadnější budoucí upravení návodů v případě nutnosti jakýchkoliv změn. Pro přípravu laboratorních návodů také není nutné pracně přepisovat informace z této práce, která je ve formátu LaTeX.

7.5 Instalační manuály

Součástí příloh je detailní postup instalace všech virtuálních systémů této laboratorní práce včetně konfigurace všech náležitostí tak, aby bylo možné laboratorní úlohy realizovat. Případné nasazení prostředí tak umožňuje využít jak již připravené systémy, tak nově vytvořené systémy ze základních šablon, nebo přímo z obrazu instalačních médií. Instalační manuály jsou v textovém formátu včetně příkazů pro instalaci.

8 ZÁVĚR

Cílem této práce bylo nastudování problematiky HTTPS, identifikace útoků vhodných pro laboratorní práce a implementace těchto útoků do virtuálního prostředí VMware. Úkoly jsou koncipovány tak, aby je bylo možné využít jako samostatné laboratorní práce. V souvislosti s virtuálním prostředím bylo nutné implementovat vhodné skripty a zranitelné systémy využitelné v laboratorních úlohách. Následně bylo nutné navrhnout scénář útoků tak, aby je bylo možné realizovat během doby laboratorních cvičení.

V práci byly realizovány čtyři laboratorní úlohy na různé oblasti problematiky zranitelností HTTPS. Každá z těchto úloh řeší jinou problematiku, což by mělo vést k větší atraktivitě zadání pro studenty. Úlohy popisují jednotlivé problematiky z více úhlů pohledu. Po nastudování teoretických prerekvizit student zkoumá cílový systém z pohledu útočníka, který se snaží nalézt jeho zranitelnost. Dalším krokem je samotný útok, který vede k úspěšnému získání očekávaných aktiv v podobě cookie, soukromého klíče nebo znalosti hesla. V posledním kroku se student z pohledu správce snaží identifikovat možná opatření k částečnému nebo úplnému zamezení zranitelnosti umožňující daný útok.

Praktické části využívají tří typů počítačů. Klientské počítače jsou označeny předponou „cl“, počítače útočníka jsou označeny předponou „evil“a počítače, které jsou v roli serverů, jsou označeny předponou „srv“.

V první úloze se student seznámí s útokem POODLE, který znamenal závažný bezpečnostní problém protokolu SSLv3 v kombinaci s CBC. Tento útok napadá důvěrnost šifrovaného kanálu. Teoretická část je velmi detailně popsána tak, aby bylo možné zcela pochopit princip tohoto útoku. Tato úloha by mohla být považována za nejsložitější z pohledu pochopení problematiky. Z tohoto důvodu bude pravděpodobně vyžadována větší asistence ze strany vyučujícího k tomu, aby student porozuměl popisovanému problému. V praktické části byla realizace útoku záměrně zvolena ve třech krocích, které postupně co nejvíce připodobňují útok realitě, přičemž každý další krok integruje do útoku další techniky.

V druhé úloze student realizuje útok Renegotiation, který vycházel ze zranitelnosti související s opětovným vyjednáváním relace. Student v této úloze lépe pochopí souvislost mezi implementací SSL/TLS a zpracováním HTTP požadavků na straně aplikačního serveru. Tento útok napadá autentičnost a integritu zašifrovaných dat. Útočník získává možnost upravovat požadavky odesílané ze strany klienta, aniž by server zaznamenal jakýkoliv problém s bezpečností. Praktická část naznačuje využití programu Wireshark pro zjištění formátu HTTP požadavků odposlechnutím provozu na síti. Dále naznačuje další způsoby, jakými lze provádět průzkum cílového webového serveru. Konečný útok je demonstrován podvrhnutím objednávky

a modifikací jak objednaného zboží, tak adresy doručení.

Třetí úloha seznamuje studenta s útokem Heartbleed využívajícího konkrétní chyby v implementaci OpenSSL. Tato zranitelnost byla velmi závažná a díky širokému využití OpenSSL znamenala zásadní problém na poměrně velké části internetových webových serverů. Útok byl pro laboratorní úlohu zvolen i proto, že tato zranitelnost byla v minulosti považována za izolovaný problém. V rámci této úlohy by měl student pochopit, že podobné typy zranitelností vyžadují další kroky i po opravení daného problému. Útočník totiž mohl před samotnou opravou získat řadu citlivých údajů, které může i po opravě problému dále využívat. Praktická část studentovi demonstruje způsob využití obsahu paměti cílového serveru a jakým způsobem tyto informace využít k ručnímu vytvoření cookie pro neoprávněný privilegovaný přístup na cílovou aplikaci.

Ve čtvrté úloze student získá lepší povědomí o problematice certifikátu. V teoretické části si zopakuje informace týkající se šifrování a podepisování dat pomocí certifikátů a způsobů výměny klíčů používaných pro šifrování SSL. Dále se seznámí se správou certifikátů pomocí programu OpenSSL. Praktická část je rozdělena celkem na čtyři části, přičemž student by měl realizovat celkem tři části – první část, jednu ze dvou následujících částí (tedy druhou nebo třetí část) a vždy čtvrtou část. V první části je řešeno zabezpečení pfx souboru a způsobu prolomení nevhodného hesla k tomuto souboru. V praxi je běžné, že tyto pfx soubory nemají vhodně zvolená hesla a při využití pronájmu cloudového řešení pro výpočet tak může být obsah pfx souboru velice rychle dešifrován. Další dvě části se zabývají problematikou fyzického přístupu k již nepoužívaným serverům, které mohou být v řádu několika minut kompromitovány a soukromé klíče certifikátu tak lze bez větších obtíží zneužít. Čtvrtá část navazuje na předcházející kroky a poukazuje na nevhodnost využití hvězdičkového certifikátu, který může být po kompromitaci využit k odposlechnutí dalších systémů využívajících tento certifikát.

Součástí práce bylo mimo jiné vypracování příruček pro učitele ve formátu pdf a docx, dále pak kompletní infrastruktura certifikační autority pro generování nových certifikátů podepsaných touto autoritou, podrobné instalační návody nebo bash skripty pro vytvoření všech využívaných virtuálních počítačů v laboratorních úlohách a také laboratorní úlohy ve formátu pdf a docx.

Na základě výše uvedeného tedy mohu konstatovat, že jsem dosáhl cílů stanovených v této práci.

Realizace prostředí laboratorních úloh byla náročná z důvodu nalezení vhodných zranitelností a realizací scénářů jednotlivých úloh. Úlohy byly voleny tak, aby bylo možné praktickou část dokončit v době stanovené pro laboratorní cvičení. Velmi pro-

blematická byla také konfigurace virtuálního prostředí, které muselo obsahovat často již nepoužívané verze softwaru s danými zranitelnostmi. I přes nalezení vhodných skriptů obnášelo spuštění skriptů další dlouhodobou analýzu kódu a prerekvizit, které nebyly vůbec, nebo jen částečně, zdokumentovány. Pro úspěšnou realizaci a pochopení vybraných útoků tak bylo nutné časově náročné zpětné inženýrství a analýza šifrovaného síťového provozu. Další komplikací při instalaci systémů bylo zamezení automatických aktualizací, které by na systémech opravily dané zranitelnosti. To následně znemožnilo využití přímo dostupných repozitářů s balíčky a při výzkumu bylo nutné využívat jednotlivé starší verze balíčku, popř. využívat staré neopravené zdrojové kódy.

Při porovnání všech útoků v této práci je z pohledu rizika nejvíce vnímán útok Heartbleed, který při malých časových nárocích umožňuje útočníkovi získat velmi citlivé údaje. Je zřejmé, že podobné útoky založené na chybném způsobu implementace při programování mohou být i do budoucna velmi kritické. Vývoj aplikací souvisejících s SSL/TLS by měl být lépe prověřován, aby byl výskyt podobných zranitelností minimalizován. Jako druhý nejkritičtější útok je vnímáno zneužití soukromého klíče, přičemž tomuto útoku lze předejít vhodným zacházením s certifikáty a také důsledným auditováním správy klíčů. Ke snížení rizika by velkou měrou přispělo šifrování dat na discích serverů.

Během zpracování této práce bylo identifikováno několik oblastí, na kterých by mohl být postaven další výzkum.

Při vyhledávání zranitelných systémů byla identifikována distribuce obsahující záměrně velké množství zranitelností, a to nejen z oblasti HTTPS. Tato distribuce byla, i když pouze okrajově, využita také v laboratorních úlohách. Distribuce beebox [9] obsahuje solidní zázemí pro další laboratorní práce zaměřené na „ethical hacking“, které by mohly být využity v oborech zaměřených na bezpečnost.

Další výzkum by mohl být proveden v souvislosti s problematikou, kterou se zabývala laboratorní úloha zaměřená na zneužití soukromého klíče. Program Wireshark zde hrál pouze pasivní roli při odposlechu komunikace, proto nebylo možné dešifrovat komunikaci sestavenou pomocí DH výměny klíčů. Obecně je znám postup útoku MITH pro DH, ovšem v případě anonymní formy. Jestliže útočník získá soukromý klíč cílového serveru, mohl by pomocí vhodně vytvořeného SSL proxy serveru realizovat útoky na další typy cipher suite, např. „DH_RSA“.

LITERATURA

[1] *AESTU. Crackpkcs12: A multithreaded program to crack PKCS#12 files. Sourceforge.net* [online]. [cit. 2017-04-07].
Dostupné z: <http://crackpkcs12.sourceforge.net/>.

[2] *APACHE.ORG. Apache HTTP Server Version 2.4: Apache Module mod_ssl. Apache.org* [online]. [cit. 2017-04-04].
Dostupné z: <http://httpd.apache.org/docs/current/mod/mod_ssl.html>.

[3] *CVE CVE-2009-3555: The Standard for Information Security Vulnerability Names* [online]. 2009 [cit. 2016-11-04].
Dostupné z: <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-3555>.

[4] *DAZZLEPOD.COM. Passwords.txt Https://dazzlepod.com* [online]. [cit. 2017-04-07].
Dostupné z: <https://dazzlepod.com/site_media/txt/passwords.txt>.

[5] *FIELDING,R.;IRVINE UC aj. RFC 2616: Hypertext Transfer Protocol – HTTP/1.1* [online]. IETF, 1999 [cit. 2016-11-04].
Dostupné z: <https://tools.ietf.org/html/rfc2616>.

[6] *FRANKS,J. aj. RFC 2617: HTTP Authentication: Basic and Digest Access Authentication* [online]. IETF, 1999 [cit. 2016-11-04].
Dostupné z: <https://www.ietf.org/rfc/rfc2617.txt>.

[7] *FREIER, A.;KARLTON, P. aj. RFC 6101: The Secure Sockets Layer (SSL) Protocol Version 3.0* [online]. IETF, 2011 [cit. 2016-11-04].
Dostupné z: <https://tools.ietf.org/html/rfc6101>.

[8] *IANA: Internet Assigned Numbers Authority* [online]. [cit. 2016-11-06].
Dostupné z: <http://www.iana.org/>.

[9] *ITSECGAMES. BWAPP: an extremely buggy web app !*[online]. [cit. 2017-03-29].
Dostupné z: <http://www.itsecgames.com/>.

[10] *JONSSON, J.;KALISKI, B. aj. RFC 3447: Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1* [online]. IETF, 2003 [cit. 2016-11-04].
Dostupné z: <https://tools.ietf.org/html/rfc3447>.

[11] *KALISKI, B. aj. RFC 2315: PKCS #7: Cryptographic Message Syntax* [online]. IETF, 1998 [cit. 2016-11-04].
Dostupné z: <https://www.ietf.org/rfc/rfc2315.txt>.

[12] *KRAWCZYK,H. aj. RFC 2104: HMAC: Keyed-Hashing for Message Authentication* [online]. IETF, 1997 [cit. 2016-11-04].
Dostupné z: <https://tools.ietf.org/html/rfc2104>.

[13] *LEE, T. Eelsivart/heartbleed.py: Heartbleed (CVE-2014-0160) Test & Exploit Python Script. Github.com* [online]. 2014 [cit. 2017-03-29].
Dostupné z: <https://gist.github.com/eelsivart/10174134>.

[14] MÖLLER, B.; THAI D.; KOTOWICZ, K. *This POODLE Bites: Exploiting The SSL 3.0 Fallback* [online]. , 4 [cit. 2016-12-02].
Dostupné z: <https://www.openssl.org/~bodo/ssl-poodle.pdf>.

[15] OPPLIGER, R. *SSL and TLS: Theory and Practice*. Switzerland: eSECURITY Technologies, 2009. ISBN 13 978-1-59693-447-4.

[16] *PATZKE, T. POODLEAttack: PoC implementation of the POODLE attack* [online]. GitHub:GitHub.com, 2017 [cit. 2017-03-25].
Dostupné z: <https://github.com/thomaspatzke/POODLEAttack>.

[17] RAY, M.; DISPENSA S.; PHONEFACTOR, INC. *Renegotiating TLS* [online]. 2009 [cit. 2016-11-04].
Dostupné z: <https://kryptera.se/Renegotiating%20TLS.pdf>.

[18] *REDTEAM PENTESTING GMBH.TLS Renegotiation Vulnerability: Proof of Concept Code.* [online]. 2009 [cit. 2017-03-26].
Dostupné z: <http://www.redteam-pentesting.de/files/tls-renegotiation-poc.py>.

[19] *RIKU,A.;RIKU, M.;GOOGLE SECURITY. HeartBleed Bug* [online]. [cit. 2016-12-02].
Dostupné z: <http://heartbleed.com/>.

[20] *ROSS, D. aj. RFC 7034: HTTP Header Field X-Frame-Options* [online]. IETF, 1999 [cit. 2016-11-04].
Dostupné z: <https://tools.ietf.org/html/rfc7034>.

[21] *SAINT-ANDRE,P. aj. RFC 6125: Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security*

(TLS) [online]. IETF, 2011 [cit. 2016-11-04].
Dostupné z: <https://tools.ietf.org/html/rfc6125>.

[22] *SHIREY,R. aj. RFC 2828: Internet Security Glossary [online]. IETF, 2000* [cit. 2016-11-04].
Dostupné z: <https://www.ietf.org/rfc/rfc2828.txt>.

[23] *SCHEIBLI, d. Iometer* [online]. [cit. 2017-04-17].
Dostupné z: <www.iometer.org>.

[24] *SCHOSSAU, J. Man in the Middle:TLS Renegotiation: pizzatime.zip* [online]. 2012 [cit. 2017-03-26].
Dostupné z: <https://sites.google.com/site/cse825maninthemiddle/tls-renegotiation/-3-howto-tutorial/pizzatime.zip>.

[25] *SRISURESH, P., JASMINE NETWORKS a K. EGEVANG. RFC 3022: Traditional IP Network Address Translator* (Traditional NAT) [online]. IETF, 2001 [cit. 2016-11-01].
Dostupné z: <https://tools.ietf.org/html/rfc3022>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

3DES	Triple Data Encryption Algorithm
AES	Advanced Encryption Standard
ARP	Address Resolution Protocol
B	Nešifrovaný blok v blokové šifře
bWAPP	buggy web application
CBC	Cipher Block Chaining, bloková šifra
CR	Carriage Return - návrat vozíku
D	dešifrování bloku
DER	Distinguished Encoding Rules
DES	Data Encryption Algorithm
DH	Diffie-Hellman
DSA	Digital Signature Algorithm
ECB	Electronic CodeBook
ECC	Elliptic Curve Cryptography
ECH	Elliptic Curve Diffie-Hellman
HMAC	Keyed-hash Message Authentication Code
HTTP	Hyper Text Transfer Protocol
HTTPS	Secured Hyper Text Transfer Protocol
IANA	Internet Assigned Numbers Authority
ID	Identifikační číslo
IEEE	Institute of Electrical and Electronics Engineer
IETF	The Internet Engineering Task Force
IIS	Internet Information Server
IMAP	Internet Message Access Protocol, mailová komunikace

IP	Internet Protocol
IV	Inicializační vektor v blokové šifře
LF	Line Feed - posun o řádek
M	blok před šifrováním, nebo po dešifrování
MAC	Medium Access Control v případě sítí, v případě šifrování Message Authentication Code
MD5	Message Digest 5
MITM	Man In The Middle
MS	Master Secret
NAT	Network Address Translation
NONCE	number used once
P7B	Public Key Cryptography Standards #7 Certificate.
PEM	Privacy Enhanced Mail Security Certificate
PFX	Personal Information Exchange
PKCS	Public Key Cryptography Standards
PMS	Premaster Secret
POODLE	Padding Oracle On Downgraded Legacy Encryption
PRF	Pseudo Random Function
RC4	Rivest Cipher 4
RFC	Request For Comment
SHA-1	Secure Hash Algorithm 1
SSD	Solid State Disc
SSL	Protokol Secure Socket Layer
TCP	Transmittion Control Protocol
TLS	Transport Layer Security

UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XOR	Exlusive OR

SEZNAM PŘÍLOH

A OBSAH PŘILOŽENÝCH BLU-RAY

První přiložený disk obsahuje hlavní soubory této práce.

Adresář **CA** obsahuje zip archiv obsahující celou konfiguraci certifikační autority, kterou lze využít pro případné vygenerování dalších certifikátů.

Adresář **Diplomová_práce** obsahuje všechny soubory nutné pro rekompilaci celého dokumentu práce.

V adresáři **Instalace** jsou uloženy podrobné postupy, jak lze z šablon virtuálních počítačů nakonfigurovat jednotlivé počítače využité v laboratorních úlohách.

V adresáři **Laboratorní_úlohy** jsou uloženy dokumenty využitelné jako návody pro studenty. Tyto soubory jsou identické s obsahem laboratorních úloh v tomto dokumentu, pouze jsou pro snadnější úpravu uloženy ve formátu docx. V podsložkách jsou pak tyto docx soubory uložené také v pdf formátu.

V adresáři **Příručky_vyučující** jsou uloženy příručky pro vyučující ve formátu docx. V podsložkách jsou pak tyto docx soubory uloženy také v pdf formátu.

V adresáři **Virtuální_počítače** jsou uloženy zip archivy obsahující plně nakonfigurované virtuální počítače využívané v laboratorních úlohách.

Druhý přiložený disk obsahuje dodatečné soubory této práce.

Adresář **Virtuální_ počítače_šablony** obsahuje šablony využitelné pro případnou novou konfiguraci virtuálních počítačů použitých v této práci. Ty šablony, které nebyly přímo staženy z internetu, lze z instalačních médií nakonfigurovat pomocí návodů v adresáři Instalace_šablon. V adresáři není přiložena šablona pro počítač s operačním systémem Windows 2012 R2, tato šablona musí být nakonfigurována podle návodu v adresáři Instalace_šablon.

Adresář **Obrazy** obsahuje obrazy instalačních médií nutných pro konfiguraci šablon virtuálních počítačů, které nebyly přímo staženy z internetu.

Adresář **Instalace_šablon** obsahuje postupy konfigurace šablon virtuálních počítačů, které nebyly nebyly přímo staženy z internetu.

- `/` ..... kořenový adresář prvního přiloženého disku
  - `CA` ..... adresář se zip archivem certifikační autority
  - `Diplomová_práce` ..... adresář se zdrojovými soubory práce
    - `Snímky_laboratorní_úlohy` ..... png snímky jednotlivých úloh
      - `Heartbleed`
      - `Poodle`
      - `TLS_renegotiation`
      - `Zneužití_soukromého_klíče`
    - `Vektorová_grafika_křivky` ... svg soubory po konverzi textu na křivky
    - `Vektorová_grafika_text` .... svg soubory před konverzi textu na křivky
    - `Zdrojové_kódy_latex` ..... TeXnicCenter projekt se všemi soubory
      - `loga` ..... loga využitá v práci
      - `obrazky` ..... obrázky využité v práci
      - `pdf` ..... pdf soubory vložené do práce
      - `text` ..... texty práce
  - `Instalace` ..... adresáře s instalačními instrukcemi jednotlivých virtuálů
    - `cl1` ..... instalační shell skripty včetně souborů pro instalaci cl1
    - `cl2` ..... instalační shell skripty včetně souborů pro instalaci cl2
    - `evil1` ..... instalační shell skripty včetně souborů pro instalaci evil1
    - `evil2` ..... instalační shell skripty včetně souborů pro instalaci evil2
    - `srv1` ..... instalační shell skripty včetně souborů pro instalaci srv1
    - `srv2` ..... instrukce pro instalaci srv2
    - `srv3` ..... instalační shell skripty včetně souborů pro instalaci srv3
    - `srv4` ..... instalační shell skripty včetně souborů pro instalaci srv4
  - `Laboratorní_úlohy` ..... návody k laboratorním úlohám
    - `docx` ..... návody pro studenty v docx souborech
    - `pdf` ..... návody pro studenty v pdf souborech
  - `Příručky_vyučující` ..... příručky pro vyučující
    - `docx` ..... příručky pro vyučující v docx souborech
    - `pdf` ..... příručky pro vyučující v pdf souborech
  - `Virtuální_počítače` .... adresáře se zip archivy obsahující nakonfigurované počítače

- `/` ..... kořenový adresář druhého přiloženého disku
  - `Virtuální_počítače_šablony` ... adresář se zip archivem virtuálních šablon
  - `Obrazy` ..... obrazy instalačních médií
  - `Instalace_šablon` ..... instalace virtuálních šablon z instalačních obrazů