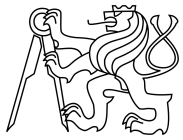
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Zabezpečení komunikace s učebnovými PC
Student:	Tomáš Šmíd
Vedoucí:	Ing. Ladislav Vagner, Ph.D.
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Při vypracovávání zkoušek na PC je potřeba zabezpečit komunikaci mezi učebnovým PC a vyhodnocovacím systémem (např. Progtest). Je potřeba zejména zajistit, aby v době konání zkoušky mohli své výsledky odevzdávat pouze studenti přítomní ve zkušební místnosti.

1. Analyzujte stávající řešení zabezpečení komunikace pomocí nástroje stunnel.
2. Navrhněte a implementujte vylepšené řešení postavené na myšlence ssl-to-ssl proxy.
3. Integrujte implementovanou proxy do systému ProgtestEnv, kterým se v současnosti řídí zabezpečení učebnových PC.
4. Otestujte implementované řešení.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 14. února 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra Softwarového inženýrství

Bakalářská práce

Zabezpečení komunikace s učebnovými PC

Tomáš Šmíd

Vedoucí práce: Ing. Ladislav Vagner, Ph.D.

15. května 2017

Poděkování

V první řadě bych rád poděkoval všem lidem, kteří mi dali možnost nebo mi přímo pomohli s prohlubováním praktických znalostí týkajících se vývoje počítačových aplikací a bezpečnosti v informatice.

Dále bych rád poděkoval Ing. Ladislavu Vagnerovi, Ph.D., vedoucímu mé bakalářské práce, za jeho vstřícný přístup a pomoc při analýze současného řešení.

Děkuji Michaele Onuferové a Tereze Šmídové za pomoc s hledáním chyb v textu této práce.

Taktéž děkuji své rodině a nejbližšímu okolí za podporu při dosavadním studiu a v životě.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 15. května 2017 .....

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Šmíd, Tomáš. *Zabezpečení komunikace s učebnovými PC*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017. Dostupný také z WWW: ⟨`https://bitbucket.org/Biscuide/radford_proxy`⟩.

Abstrakt

Tato bakalářská práce se zabývá analýzou a návrhem vylepšeného řešení zabezpečení komunikace mezi učebnovými počítači a vzdáleným vyhodnocovacím systémem na Fakultě informačních technologií ČVUT. Práce dále popisuje autentizaci uživatelského prostředí na dálku. Bezpečnost komunikace bude zajištěna kryptografickým protokolem TLS a výsledná aplikace bude postavena na knihovně OpenSSL.

Klíčová slova proxy, HTTPS, SSL/TLS, PKI, stunnel, OpenSSL, zabezpečení komunikace, autentizace, Progtest, Certifikát, C++

Abstract

This bachelor's thesis describes the analysis and design of improved communications security between classroom's computers and remote evaluate system in the FIT CTU. Thesis also discusses remote authentification of user's environment. Communications security will be provided by TLS cryptographic protocol and final application will be based on OpenSSL library.

Keywords proxy, HTTPS, SSL/TLS, PKI, stunnel, OpenSSL, communications security, autentization, Progtest, Certificate, C++

Obsah

Úvod **1**

1 Cíl práce **3**

2 Analýza **5**
2.1 Současné řešení 5
2.2 Funkční požadavky 7
2.3 Nefunkční požadavky 7
2.4 Identifikační data 7
2.5 Rizika 9
2.6 Základní charakteristika výstupní aplikace 9

3 Návrh **11**
3.1 Návrh tříd 11
3.2 Inicializace 23
3.3 Identifikace proxy 25
3.4 Bezpečnost 26
3.5 Nasazení 27
3.6 Použité nástroje a technologie 28

4 Implementace a realizace **29**
4.1 Načtení vstupních dat 29
4.2 Konfigurace 31
4.3 Vytvoření spojení a správa session 31
4.4 Zpracování a přeposlání příchozích dat 32
4.5 Nastavení certifikátů proxy 32
4.6 Vlastní ověření certifikátu vzdáleného serveru 34
4.7 Zavedení do startu systému 34
4.8 Vystavení certifikátu pro proxy 35

5 Testování **39**
5.1 Vstupní nastavení 39
5.2 Testovací scénáře 39

Závěr **43**

Literatura **45**

A Seznam použitých zkratek **49**

B Obsah přiloženého CD **51**

C Přiložené obrázky **53**

D Přiložené ukázky **57**

Seznam obrázků

2.1 Schéma současného řešení 6

3.1 ISO/OSI model 19
3.2 TLS v ISO/OSI modelu 20
3.3 TLS handshake 20
3.4 Třída Session – Rozdělení prostředků 22
3.5 Návrh modelu nasazení 27

4.1 Třída HttpClient – Diagram funkce ResendRequest 33

5.1 Pozitivní testovací scénář 40

C.1 Fronta producent-konzument 54
C.2 UML diagram tříd (výřez) 55

Seznam ukázek

3.1 HTTP zpráva – základní dělení obsahu 16
3.3 HTTP požadavek na server progtest.fit.cvut.cz 17
3.2 HTTP zpráva (požadavek) – formát prvního řádku 17
3.4 HTTP zpráva (odpověď) – formát prvního řádku 17
3.5 HTTP odpověď ze serveru progtest.fit.cvut.cz 18
4.1 Výřez z konfiguračního souboru radford.cfg 30
4.2 Zdrojový kód funkce ProxyHttp::Accept 32
4.3 Informace o koncovém certifikátu pro proxy 37
D.1 Zdrojový kód funkce Start – vybírání objektů Session z fronty . 57
D.2 Kořenový certifikát v PEM formátu 58

Úvod

Na Fakultě informačních technologií ČVUT (dále jen FIT ČVUT) jsou v současnosti v mnoha předmětech ověřovány znalosti studentů pomocí vzdáleného vyhodnocovacího systému. V případě úloh, jejichž čas a místo na vypracování jsou vyhrazeny ve škole (např. zkouška), mohou být studenti pozváni k počítačům nacházejících se přímo v učebnách budovy FIT ČVUT. V čase řešení těchto úloh je nezbytné zajistit, aby se student na počítači pohyboval pouze v povolených prostředích. Tato prostředí mnohdy vytvářejí další omezení pro studenty jako je například elektronická komunikace s okolním světem. Z tohoto důvodu je nutné zajistit, aby komunikace mezi počítačem používaným studentem a vzdáleným serverem byla nejen zabezpečena, ale taktéž aby bylo možné na vzdáleném serveru ověřit původ příchozí komunikace, tj. zdali data, která jsou součástí této komunikaci, pochází pouze z povolených prostředí.

Dnešní řešení správy komunikace mezi počítačem a vyhodnocovacím systémem je postaveno na nástroji stunnel[1]. Jedná se o *open source* program třetí strany. Toto řešení není zcela vyhovující po stránce technického provedení a to zejména z důvodu komplikovanosti.

Nové řešení bude postaveno na technologii protokolu TLS a softwarovém proxy serveru. Proxy servery jsou v dnešní době hojně využívány jako prostředník mezi klientem a serverem. Účely jejich použití se mohou lišit, v případě této práce se bude jednat o webový HTTPS proxy server. Úkolem proxy bude mimo jiné do HTTP požadavků doplňovat dodatečná metadata. Tato metadata budou obsahovat identifikační informace včetně jejich elektronického podpisu, který bude možné následně zkontrolovat na straně cílového serveru.

V první kapitole se věnuji analýze současného řešení postaveného na nástroji stunnel v prostředí *ProgtestEnv*, popisu samotného prostředí *ProgtestEnv* a požadavkům na výslednou aplikaci. Druhá kapitola obsahuje návrh aplikace proxy serveru včetně případných doplňujících informací k implementaci. Součástí třetí kapitoly je popis a náhled na výslednou implementaci proxy serveru.

[1] https://www.stunnel.org/index.html.

Poslední kapitola se zabývá samotným testováním výstupní aplikace.

Texty v této práci týkající se problematiky kryptografie, jmenovitě charakteristiky kryptografických objektů, ověření certifikátů, ověření a vytvoření elektronických podpisů aj. jsou mnohdy opřeny o vlastní znalosti a zkušenosti. Hlavním zdrojem nabytí těchto vědomostí byl mimo jiné literární zdroj [1].

Kapitola 1

Cíl práce

Výstupem této bakalářské práce je návrh a implementace vylepšeného řešení zabezpečení a identifikace komunikace mezi učebnovými počítači a vzdáleným vyhodnocovacím systémem na Fakultě informačních technologií ČVUT. Práce má za účel nahradit stávající řešení, jehož součástí je nástroj stunnel. Vylepšené řešení bude zhotoveno v podobě softwarového proxy serveru, který bude součástí testovacího prostředí *ProgtestEnv*. Výstup práce bude též obsahovat zohlednění návrhu systému pro autentizaci uživatelského prostředí (operačního systému), na kterém bude proxy server spuštěn. Cílem práce je vytvořit jednodušší a flexibilnější systém pro přenos dat mezi počítačem a vzdáleným serverem s důrazem na zachování bezpečnosti a zjednodušení provedení po technické stránkce.

Kapitola 2

Analýza

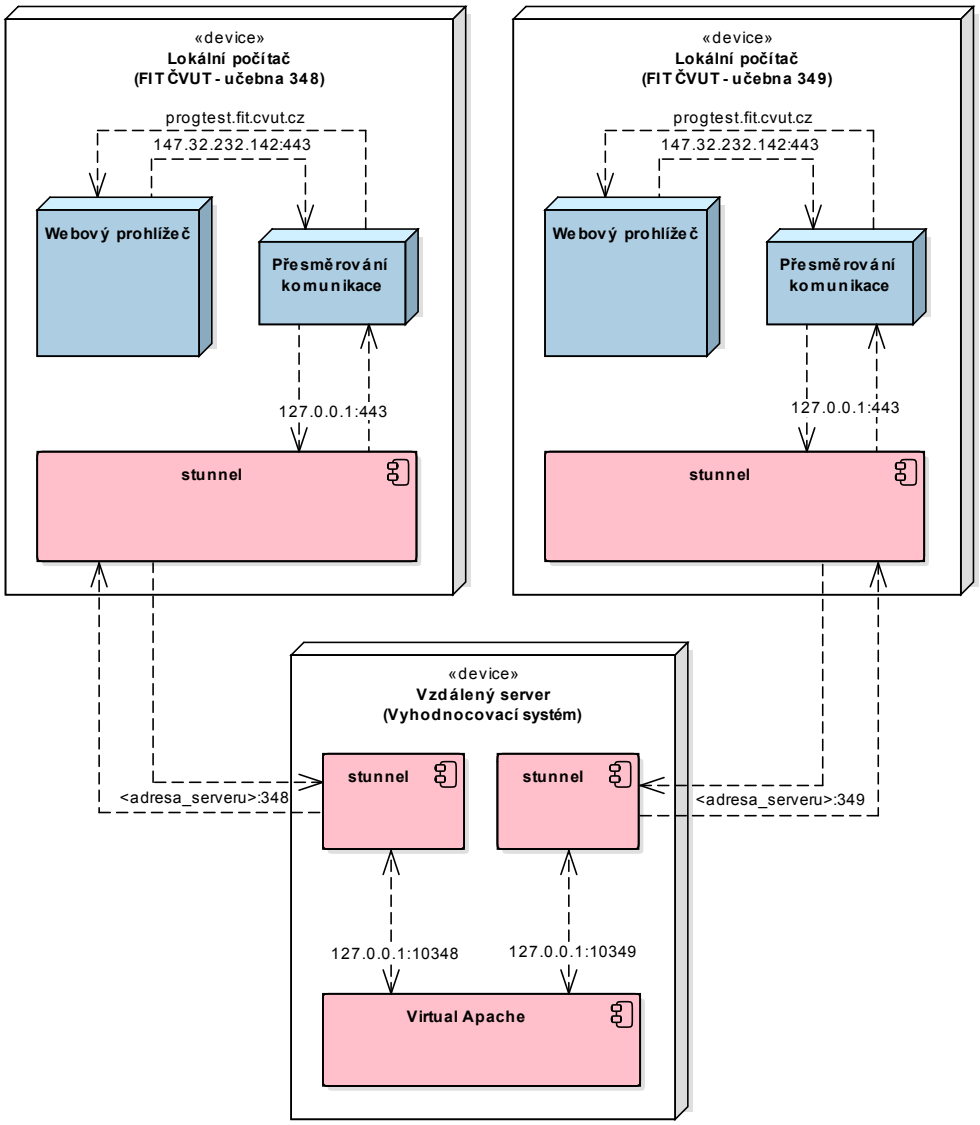
Obsahem kapitoly jsou informace o současném řešení, které jsem získal v průběhu komunikace s vedoucím této práce Ing. Ladislav Vagner, Ph.D. Kapitola dále zahrnuje analýzu funkčních a nefunkčních požadavků, analýzu rizik a základních charakteristik výstupní aplikace.

2.1 Současné řešení

Součástí současného řešení je nástroj stunnel, který je spouštěn na lokálním počítači v testovacím prostředí *ProgtestEnv* (více viz 2.1.1). Nástroj stunnel naslouchá na *loopback* adrese 127.0.0.1 s portem 443. Zajišťuje otevření TLS tunelu ke vzdálenému vyhodnocovacímu systému a přenos dat skrze tento tunnel. Použití nástroje stunnel však není zcela ideálním řešením pro danou problematiku z pohledu technické stránky. Jedná se zejména o následující důvody:

- **Problém s identifikací příchozích požadavků do vyhodnocovacího systému** – Vyhodnocovací systém nemá možnost jednoduše rozeznat, ze které učebny pochází příchozí požadavek. Navíc nedokáže plnohodnotně ověřit zdroj a čas vytvoření takového požadavku. V současném řešení je rozeznání učebny založeno pouze na způsobu otevření tunelu (skrze nástroj stunnel) a to tak, že otevření tunelu probíhá na straně vzdáleného vyhodnocovacího systému na specifickém portu pro danou učebnu. To znamená, že vyhodnocovací systém musí naslouchat na unikátním portu pro každou učebnu. Ve vyhodnocovacím systému následně dochází k *loopback* přesměrování portů na požadovanou cílovou adresu.

- **Dvojnásobné šifrování komunikace** – Řešení s použitím nástroje stunnel poskytuje přenos dat skrze tunel, který je šifrovaný protokolem TLS. Avšak samotná data, která jsou přenášena, jsou součástí protokolu HTTPS, který taktéž používá pro šifrování dat protokol TLS. Z tohoto



Obrázek 2.1: Schéma současného řešení

důvodu dochází k dvojnásobnému použití stejného šifrovacího algoritmu a zbytečné nevyžádané režii navíc.

Schéma zachycující stav současného řešení je součástí obrázku 2.1.

2.1.1 Prostředí ProgtestEnv

Testovací prostředí *ProgtestEnv* je postaveno na linuxové distribuci Gentoo. Prostředí je složeno ze sady skriptů, které upravují nastavení samotného načte-

ného obrazu operačního systému. Skripty jsou prováděny v příslušném logickém pořadí při bootu OS. Tyto skripty zajišťují požadované nastavení testovacího prostředí a jejich součástí je například omezení elektronické komunikace s okolním světem (včetně interní sítě) nebo odstranění modulu pro načtení externích přenosných úložišť (USB klíč, paměťová karta, ...).

2.2 Funkční požadavky

Následující seznam obsahuje funkční požadavky definované zadavatelem této práce:

- Aplikace musí podporovat protokol HTTPS.
- Aplikace musí umět vložit dodatečná identifikační data do hlavičky požadavků.
- Aplikace musí umět podepsat dodatečná identifikační data v hlavičce.
- Aplikace musí umět ověřit certifikát cílového serveru.
- Aplikace se musí umět prokázat svým certifikátem klientovi.

2.3 Nefunkční požadavky

Obsahem sekce je výčet nefunkčních požadavků na aplikaci.

- Aplikaci musí být možné spustit na operačním systému distribuce Linux, konkrétně se jedná o distribuci Gentoo, na němž je postaveno testovací prostředí *ProgtestEnv*.
- Běžný uživatel počítače, na kterém poběží aplikace, nesmí mít přístup k této aplikaci a jejím konfiguračním a inicializačním souborům.
- Aplikace bude podporovat omezenou množinu domén.
- Aplikace bude pro kryptografické operace používat ověřenou kryptografickou knihovnu.

2.4 Identifikační data

Problém s identifikací příchozích požadavků lze vyřešit tak, že do HTTP zpráv odcházejících z klienta budou doplněna identifikační data, která budou sloužit pro jednoznačné určení původu zpráv. Z analýzy vyplývá, že pro identifikaci klientské strany budou dostačující následující informace:

1. časový údaj o vytvoření HTTP zprávy,

2. identifikátor učebny,

3. podpis výše uvedených informací(viz 2.4.1).

Pro identifikaci konkrétního počítače bude sloužit IP adresa, ze které požadavek přišel. Tyto všechny identifikační údaje budou součástí příchozích HTTP zpráv na vzdálený vyhodnocovací systém. Vzdálený systém je nadále bude zpracovávat a může s nimi nakládat dle své interní logiky.

2.4.1 Elektronický podpis

Vytvořený podpis identifikačních dat musí minimálně zajišťovat autentifikaci a integritu podepsaných dat. Vzhledem k faktu, že počet požadavků může být velmi vysoký, je nutné vybrat takový algoritmus, který bude efektivní jak pro vytvoření samotného podpisu, tak i pro jeho ověření na vzdáleném serveru. Vhodným kandidátem je skupina algoritmů z rodiny MAC (*Message Authentication Code*).

2.4.1.1 TPM

Součástí původního řešení bylo použití kryptografického čipu TPM sloužícího k autentifikaci. Tento čip umožňuje mimo jiné použití asymetrických algoritmů pro podpis a ověření dat. Bezpečnost použití toho čipu je založena na faktu, že veškeré privátní klíče, které jsou součástí tohoto čipu, nelze jakkoliv získat (na rozdíl od veřejných klíčů). Použitím privátního klíče z čipu TPM k elektronickému podpisu dat by tudíž došlo k jednoznačné identifikaci podepisovatele.

V průběhu návrhu o implementaci se však vyskytly problémy. Některé z těchto problémů nebyly v mnoha případech „rozumně“ řešitelné. Jedná se o následující záležitosti:

- nekompatibilita mezi jednotlivými verzemi čipu TPM,
- nekompatibilita mezi čipy TPM od jednotlivých výrobců,
- problém s použitím samotného čipu TPM (nedostatek informací),
- ověřením podpisu by došlo výhradně k ověření hardware počítače, nikoliv software (OS a případně proxy).

Z důvodu výše uvedených bylo nakonec ustoupeno z požadavku na použití kryptografického čipu TPM a muselo být hledáno jiné řešení (viz subsekce 3.2.3), které by sloužilo k autentizaci systému.

2.5 Rizika

2.5.1 Souborový systém

Binární soubor aplikace proxy serveru bude společně s konfiguračním souborem uložen v takové cestě souborového systému, kam běžný uživatel nebude mít oprávnění přístupu. Avšak vzhledem ke skutečnosti, že celý testovací obraz operačního systému je veřejně dostupný, je nutné počítat s tím, že se aplikace společně s konfiguračním souborem může dostat do rukou útočníka.

Všechny ostatní soubory, které budou použity pro inicializaci proxy serveru v testovacím prostředí, nebudou útočníkovi dostupné z důvodu jejich načtení nebo stažení až v průběhu konfigurace OS. Útočník by tak nejdříve musel získat vyšší oprávnění, než mu náleží – přihlásit se jako jiný uživatel s vyššími právy nebo obejít kernel systému. V takovém případě se jedná o daleko závažnější útok. Vzhledem k tomuto faktu nemá útočník možnost jakkoliv běžnou cestou podvrhnout svoje (škodlivé) soubory.

2.5.2 Aplikace

S přihlédnutím ke skutečnosti, že binární podoba aplikace proxy serveru je veřejně dostupná (viz předchozí subsekce), může útočník disasemblovat tento binární soubor a snažit se v něm dohledat slabé místo. Slabé místo se může například nacházet ve vykonávání kódu pro zpracování příchozí a odchozí komunikace. Tuto nalezenou slabinu lze následně použít pro útok (např. změna dat). Útočník by též mohl dohledat místo v paměti procesu, kam se načítá sdílené tajemství, které je používáno pro identifikaci (viz 3.3).

2.5.3 Autentizace

Autentizace a správa inicializačních dat bude probíhat skrze externí aplikaci (Wrapper – viz sekce 3.2), která není součástí této bakalářské práce. Privátní klíč, který bude sloužit k autentizaci, bude distribuován pomocí externího úložiště (např. USB klíč). Privátní klíč bude součástí souborového systému jenom do doby, než dojde k autentizaci a stažení inicializačních dat pro proxy. K privátnímu klíči uloženému v systémovém souboru se útočník nedostane, neboť bude používán pouze v době konfigurace OS za přítomnosti oprávněné osoby. Útočník však může využít nepozornosti vlastníka USB klíče a zcizit tento USB klíč s veškerým celým jeho obsahem. V takovém případě je nutné neprodleně kompromitovat veřejný klíč v páru s privátním klíčem na zcizeném USB klíči.

2.6 Základní charakteristika výstupní aplikace

Výstupní aplikace by měla umět zpracovávat protokol HTTPS. Do tohoto zpracování spadá práce s protokolem TLS – rozšifrování příchozí komunikace

a opětovné zašifrování komunikace odchozí.

Aplikace by měla být navržena tak, aby podporovala minimálně protokoly HTTP a HTTPS. Z toho vyplývá, že řízení protokolů by mělo spadat pod obecné rozhraní a že součástí implementace by měla být třída, která bude spravovat tyto jednotlivé protokoly. Tento bod je opřen o fakt, že protokol HTTPS je nadstavbou protokolu HTTP.

Součástí výsledné aplikace by též měla být externí konfigurace, která umožní jednoduché a zároveň flexibilní nastavení jednotlivých vstupních parametrů při startu procesu. Dostupnost externí konfigurace je z důvodu, aby nebylo nutné pro každé rozdílné nastavení vytvářet nový binární soubor aplikace.

Další nedílnou součástí je logovací systém, který umožní ukládat data a informace z běžícího procesu (například popis chyb). Tento logovací systém by měl být schopen zapisovat jak na standardní (případně chybový) výstup, tak do souboru.

Vzhledem k charakteristice požadavků na funkcionalitu softwarového proxy serveru bude výsledná aplikace navržena jako vícevláknová. To znamená, že součástí návrhu musí být systém, který bude zajišťovat logiku paralelního zpracování mnoha spojení najednou. Paralelní zpracování více požadavků plyne z faktu, že webové prohlížeče v dnešní době běžně otevírají více než jedno spojení na vzdálený server.

Kapitola 3

Návrh

Zadání práce neobsahuje specifikaci požadovaných technologií nebo vzorů, které jsou upřednostňovány nebo požadovány pro návrh a implementaci softwarového proxy serveru.

Na základě analýzy bude výsledná aplikace vícevláknová. Z toho vyplývá, že každé zpracované příchozí spojení bude mít k dispozici část systémových prostředků, které budou přiřazeny proxy serveru. Pro vyřizování požadavků od připojených klientů bude zvolen vzor producent-konzument, aby nedošlo k zahlcení prostředků systému nebo samotného procesu proxy serveru. Vzhledem k použití zmiňovaného vzoru bude docházet k asynchronnímu zpracování připojení klientů, přičemž počet konzumentů a délka fronty čekajících klientů budou omezeny nastavením proxy serveru.

Proxy server bude společně s externí aplikací Wrapper spouštěn při bootu operačního systému. Po spuštění aplikace Wrapper bude následovat autentizace a inicializace. V případě, že dojde k chybě v inicializační části, nedojde k vykonávání kódu, který zajišťuje přeposílání dat z lokálního počítače na server neboli proxy nebude fungovat. O inicializaci vstupních dat pro proxy server se bude starat externí proces Wrapper. Použití a popis kladených funkčních požadavků na tuto aplikaci je součástí sekce 3.2.

3.1 Návrh tříd

Navržené třídy jsou rozděleny do několika bloků. Tyto bloky tvoří samostatné logické oddíly. Způsob návrhu zajišťuje v případě potřeby do budoucna flexibilnější rozšíření jednotlivých bloků. Podstatná část objektového návrhu v podobě UML diagramu tříd je součástí přílohy C.2. Kompletní UML diagram tříd je součástí obsahu na přiloženém CD.

3.1.1 Crypto

Třídy obsažené v subsekci s názvem *Crypto* budou sloužit výhradně k obsluze krypografických objektů, které jsou potřebné například pro ověření certifikátu cílového serveru nebo zprostředkování vlastního certifikátu proxy serveru pro koncového uživatele. Tyto třídy budou výhradně používány v protokolu HTTPS. Bude se jednat o nadstavbové objekty nad strukturami definovanými v knihovně OpenSSL. Objekty budou mít za účel poskytovat API pro zjednodušení operací nad strukturami z knihovny OpenSSL.

3.1.1.1 Certificate

Součástí třídy *Certificate* bude struktura typu X.509v3, která definuje podobu certifikátu používaného ve standardu PKI. Objekt X.509v3 obsahuje mnoho položek obsáhujících různé informace. Následující výjmenované položky jsou jedny z klíčových pro tuto práci:

- **Subject** – Název subjektu (ve formátu *X.500 Distinguished Names*). V případě protokolu HTTPS se jedná o nezbytný identifikátor pro určení domény, pro kterou byl koncový certifikát vydán.
- **Subject public key** – Obsahuje veřejný klíč subjektu a identifikátor algoritmu, který tento klíč používá. Veřejný klíč patří společně s privátním klíčem do páru k danému certifikátu. Používá se pro ověření dat, která byla podepsána privátním klíčem. Privátní klíč není součástí struktury X.509v3.
- **Issuer** – Název vydavatele (ve formátu *X.500 Distinguished Names*). Společně s položkou *The Authority Key Identifier* (AKID) slouží pro identifikaci vydavatele certifikátu. Tato identifikace se využívá například pro sestavení certifikační cesty ke kořenovému certifikátu (více o identifikaci certifikátů v průběhu procesu sestavení certifikační cesty viz [2]).
- **Extensions** – Položka, která byla přidána až ve standardu verze 3 objektu X.509. Definuje například způsoby použití klíče (např. šifrování, podepisování), certifikační politiky (např. povinné položky certifikátu, naplnění položek certifikátu) a jiná omezení (např. délka certifikační cesty, způsob použití certifikátu).
- **Validity** – V položce *Validity* je uložena informace o intervalu, ve kterém je certifikát platný. Počátek intervalu (čas počátku platnosti) je značen hodnotou notBefore a konec intervalu (čas konce platnosti) je značen notAfter. *Obě tyto hodnoty mohou být uloženy ve tvaru UTCTime nebo GeneralizedTime* [3].
- **Signature** – Obsahuje hodnotu podpisu certifikátu a identifikátor algoritmu, kterým byl podpis vytvořen. Hodnota podpisu se používá pro

ověření, zdali byl certifikát podepsán privátním klíčem vydavatele. Kontrola hodnoty podpisu certifikátu zajišťuje, zdali nebyl certifikát podvržen útočníkem.

Více informací o struktuře a použití objektu X.509v3 se lze dočíst ve standardu RFC 5280 [3]. Účelem třídy *Certificate* bude zjednodušení práce s certifikáty v modelu PKI. Příklad strukturovaného výpisu základních informací, které jsou obsaženy v certifikátu, lze najít v ukázce 4.3. Požadavky na certifikát, kterým se bude prokazovat proxy server, jsou součástí sekce 4.8.

3.1.1.2 PrivateKey

Třída *PrivateKey* bude obsahovat strukturu typu EVP_PKEY, která slouží k uložení veřejného a privátního klíče v asymetrické kryptografii. V případě této třídy bude struktura naplněna pouze privátním klíčem. Uložené klíče v této třídě mohou být mnoha formátů, za zmínění stojí formáty pro algoritmy ECDSA a RSA[2] (více viz [4]). Zmíněné typy klíčů jsou hojně používány mimo jiné v protokolu TLS. Klíč ECDSA je variantou privátního klíče, který je založen na eliptických křivkách. Vzhledem k faktu, že eliptické křivky mají v kryptografii do budoucna světlejší předpověď než algoritmus RSA, a to zejména díky menší velikosti klíčů se stejným nebo vyšším počtem bitů zabezpečení a vyšší rychlosti podpisu dat, je tato podpora použití různých typů klíčů vítána [5].

Třída bude použita například pro uložení privátního klíče, který patří do páru s veřejným klíčem certifikátu proxy serveru. Tímto certifikátem se bude proxy prokazovat klientovi. Privátní klíč certifikátu bude používán k šifrování odchozí komunikace ke klientovi, resp. rozšifrované příchozí komunikace ze serveru.

3.1.2 Framework

Součástí subsekce *Framework* jsou třídy, které budou používány skrze celou aplikaci. Tyto třídy mohou mít spíše podpůrný nebo doplňkový účel.

3.1.2.1 CException

V aplikaci je vhodné mít správu chybových návratových kódů. Pro správu těchto kodů/chyb je součástí návrhu třída, která bude nahrazovat jednoduché číselné návratové kódy pomocí celého objektu. Tento objekt může v důsledku obsahovat libovolná data. Třída *CException* bude sloužit k vyvolání výjimky a bude používána pouze v případě, kdy nastane chyba v aplikaci. Tímto způsobem bude docíleno zajištění robustnější správy chybových stavů.

[2]Kryptografická asymetrická šifra, autory jsou Ron Rivest, Adi Shamir a Leonard Adleman.

Třída *CException* bude definovat podobu výjimek a jejich vytvoření. Součástí třídy budou též výčet unikátních identifikátorů typů výjimky. Tento identifikátor bude uložen v každé instanci třídy *CException*. K získání podrobnějších informací o výjimce která nastala, budou součástí třídy mimo jiné i položky obsahující informace o daném typu výjimky a situaci, při které byla daná výjimka vyvolána.

Hlavní důvod pro použití výjimek místo návratových kódů je zejména ten, že výjimky mohou, na rozdíl od číselných chybových stavů, v případě chyby nebo neočekávané situace obsahovat více potřebných informací o daném chybovém stavu.

3.1.2.2 Config

Aby aplikace mohla být jednoduše konfigurovatelná, bude potřeba zajistit výčet všech dostupných nastavení a jejich případné načtení ze souboru v pevně stanoveném formátu.

Hlavním účelem třídy *Config* bude definice všech proměnných, které mohou být nastaveny při spuštění aplikace proxy serveru. Zároveň budou definována základní nastavení jednotlivých proměnných, která budou použita při spuštění proxy serveru (pokud nebude úspěšně načten konfigurační soubor). Třída bude umožňovat načtení konfiguračního souboru. Definice formátu konfiguračního souboru je součástí subsekce 4.1.1. Obsah souboru bude parsován v průběhu inicializace konfigurace proxy serveru a v případě úspěchu načten do proměnných, které jsou součástí třídy *Config*.

Třída *Config* je navržena jako *singleton s lazy inicializací*. Tento model zaručuje, že v celém procesu proxy serveru bude existovat právě jedna instance dané třídy. Zároveň lze k této jediné instanci dané třídy usnadnit přístup, čili bude velmi jednoduše dostupná skrze celý program. Inicializace instance bude probíhat v inicializační části programu (viz sekce 3.2). Více informací o vlastnostech a použití vzoru *singleton* viz [6].

3.1.2.3 Logger

Nedílnou součástí aplikace proxy serveru bude taktéž logovací systém. Logovací systém bude umožňovat vytváření záznamů z procesu, tj. ukládat záznamy do souboru nebo odesílat na standardní výstup. Další požadovanou funkcí je základní formátování výstupu (např. přidání času).

Třída *Logger* je obdobně jako třída *Config* navržena ve vzoru *singleton s lazy inicializací*. Pro rozlišení typu záznamu jsou navrženy následující úrovně logování, které budou mít za účel zlepšit filtraci:

- **Info** – Základní úroveň pro běžné informace.
- **Warning** – Úroveň pro chyby, které nejsou závažné (aplikace může pokračovat ve vykonávání kódu v oblasti, ze které chyba pochází).

- **Error** – Úroveň pro chyby, při kterých dochází k přerušení provádění dané části kódu.
- **Debug** – Obsahuje doplňující informace ze spuštěného procesu (např. hodnoty proměnných).
- **Trace** – Určeno pro trasování volání funkcí v procesu (bude použito vždy na začátku funkce).

Vzhledem k faktu, že se jedná o návrh vícevláknové aplikace, bude nutné při implementaci vzít tuto skutečnost v potaz. To znamená, že logovací systém by měl dokázat rozeznat, ze kterého vlákna (nebo jiného objektu obsahující jednoznačný identifikátor) došlo k zavolání vytvoření záznamu. Tento požadavek na implementaci plyne z toho, aby bylo možné jednodušeji trasovat běh kódu v aplikaci pokud by bylo potřeba.

3.1.2.4 Utils

Poslední třídou subsekce *Framework* je pomocná třída nazývající se *Utils*. Jejím účelem bude zprostředkovávat různorodé funkce, které budou opakovaně používány v různých částech kódu aplikace a není je zároveň možné logicky začlenit do některého z objektů, který bude součástí implementace. S přihlédnutím k faktu, že třída by neměla mít žádný interní stav, je možné, aby jednotlivé funkce byly statické.

3.1.3 HTTP(S)

Tato část se zabývá návrhem klientů pro jednotlivé protokoly. Jedním z hlavních funkčních požadavků na aplikaci je, aby podporovala protokol HTTPS. S přihlédnutím k faktu, že protokol HTTPS je postaven na protokolu HTTP, jsou součástí návrhu oba zmiňované protokoly. Více informací o rozdílech protokolů HTTP a HTTPS se lze dočíst ve zdroji [7].

Analýza tříd pro zpracování protokolů se opírá o aktuálně nejrozšířenější standardy, přesněji:

- pro protokol HTTP se jedná o verzi 1.1,
- protokol HTTPS je postaven na HTTP verze 1.1 a TLS verze 1.2.

Výměna dat mezi klientem a serverem bude probíhat pomocí tzv. HTTP zpráv (*HTTP-message*). Přenášená data neboli zprávy se dělí do dvou skupin na základě toho, kdo je odesílatelem:

- **požadavek** (*request*) – zpráva odeslaná klientem,
- **odpověď** (*response*) – zpráva odeslaná serverem.

Výměna zpráv bude probíhat modelem požadavek-odpověď (klient odešle požadavek na server a vyčkává na odpověď od serveru). HTTP zprávy mají pevně stanovený formát standardem [8]. Základní struktura formátu je uvedena v ukázce 3.1.

```
HTTP-message =  start-line
                *( header-field CRLF )
                CRLF
                [ message-body ]
```

Ukázka 3.1: HTTP zpráva – základní dělení obsahu

Zpracováním dat z bloku *start-line* a *header-field* se zabývá subsekce 3.1.3.1 a práce s blokem *message-body* je popsána v subsekci 3.1.3.4.

3.1.3.1 HttpHeader

Třída *HttpHeader* se bude zabývat zpracováním dvou bloků z HTTP zprávy, a to blokem *start-line* a *header-field*. Logika zpracování bloku *start-line* bude dále posunuta do odvozených tříd *HttpRequest*, resp. *HttpResponse*, které jsou vedeny v subsekcích 3.1.3.2, resp. 3.1.3.3. Důvod zpracování tohoto bloku v dalších odvozených třídách plyne z rozdělení formátu bloku *start-line*. Formát se rozlišuje na základě toho, zdali se jedná o HTTP zprávu typu požadavek nebo odpověď.

Účelem této třídy bude tudíž parsovat pouze obsah bloku *header-field*. Jednotlivé položky jsou oddělené novým řádkem ve formátu CRLF[3]. Formát položek je uveden v následujícím odstavci.

„*Each header field consists of a case-insensitive field name followed by a colon (":"), optional leading whitespace, the field value, and optional trailing whitespace.*“ [8]

Parsování bloku bude probíhat na úrovni řádků, čili do instance této třídy budou ukládány páry hodnot obsahující název položky a hodnotu položky. Třída bude též poskytovat rozhraní, které umožní získat hodnotu položky z HTTP zprávy pro příslušný název položky.

3.1.3.2 HttpRequest

Účelem třídy *HttpRequest* bude parsování prvního řádku z HTTP zprávy, tj. blok *start-line*. Název bloku *start-line* je zaměněn za *request-line* z důvodu, že se jedná o zprávu typu požadavek. Formát řádku je definován normou RFC-7230 [8] (viz ukázka 3.2).

„*A request-line begins with a method token, followed by a single space (SP), the request-target, another single space (SP), the protocol version, and ends with CRLF.*“ [8]

[3]Způsob zalomení řádku.

```
GET / HTTP/1.1
Host: progtest.fit.cvut.cz
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64)
    AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.96
    Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;
Accept-Encoding: gzip, deflate, sdch, br
Accept-Language: cs-CZ,cs;q=0.8,en;q=0.6,sk;q=0.4
```

Ukázka 3.3: HTTP požadavek na server progtest.fit.cvut.cz

```
request-line = method SP request-target SP HTTP-version CRLF
```

Ukázka 3.2: HTTP zpráva (požadavek) – formát prvního řádku

Třída bude poskytovat rozhraní pro získání všech položek, které jsou obsaženy v prvním řádku HTTP zprávy. Příklad požadavku zaslaného klientem demonstruje ukázka 3.3.

3.1.3.3 HttpResponse

Obdobně jako u třídy *HttpRequest* bude třída *HttpResponse* parsovat první řádek z HTTP zprávy, avšak tentokrát pro zprávu typu odpověď. Název bloku *start-line* je též přejmenován – na *status-line*, a jeho formát je součástí ukázky 3.4.

```
status-line = HTTP-version SP status-code SP reason-phrase CRLF
```

Ukázka 3.4: HTTP zpráva (odpověď) – formát prvního řádku

Třída bude dále poskytovat rozhraní pro získání skupiny status kódu, který bude založen na položce *status-code*. Skupiny statusových kódů popisuje zdroj [9]. Příklad odpovědi ze serveru demonstruje ukázka 3.5.

3.1.3.4 HttpClient

Úkolem třídy *HttpClient* bude spravovat připojení protokolu HTTP a všechny vrstvy až po transportní (v ISO/OSI modelu, viz obrázek 3.1) pro protokol HTTPS. Pro správu připojení bude použito volání multiplatformní implementace socketu, jejíž podrobnosti jsou popsány v subsekci 3.1.6. Kromě správy připojení bude tato třída zajišťovat přenos nešifrovaných dat, tj. protokol HTTP.

```
HTTP/1.1 200 OK
Date: Fri, 05 May 2017 09:03:08 GMT
Server: Apache/2.4.10 (Debian)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
    pre-check=0
Pragma: no-cache
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 1771
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

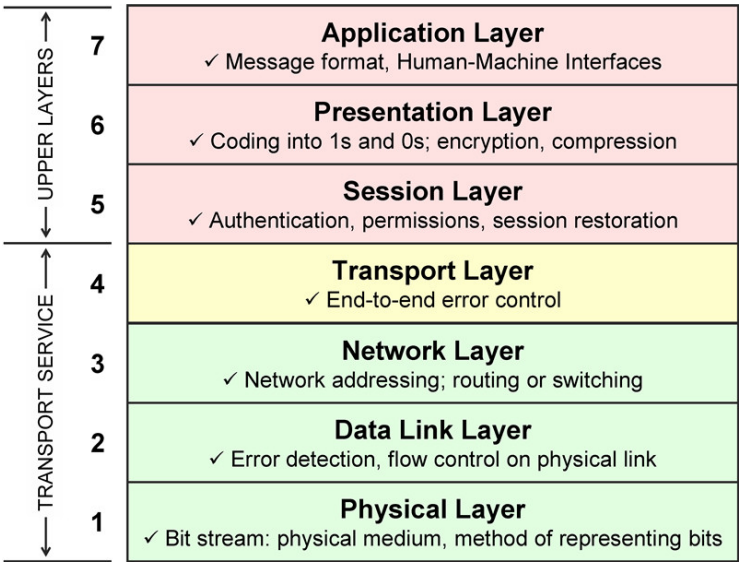
Ukázka 3.5: HTTP odpověď ze serveru progtest.fit.cvut.cz

Třída bude implementovat rozhraní jak pro přeposlání požadavku klienta, tak i pro přeposlání odpovědi ze serveru (jedná se o obdobný proces). V obou případech budou použity již zmiňované třídy *HttpRequest* a *HttpResponse*, které jsou uvedeny v subsekcích 3.1.3.2 a 3.1.3.3. Třída *HttpClient* se bude muset zároveň vypořádat s následujícími problémy:

- uzavření nebo naopak udržení spojení (na základě obsahu HTTP hlavičky a přenesených dat),
- různá kódování obsahu bloku *message-body*,
- timeout spojení.

S příchodem protokolu HTTP verze 1.1 byla navíc standardizována tzv. perzistentní spojení. Tato spojení jsou charakteristická tím, že v rámci jednoho spojení může sériově proběhnout více požadavků a následných odpovědí. Jedná se o způsob, jak ušetřit čas na opakovaném otevírání a uzavírání spojení mezi klientem a serverem. Informace o tom, zdali se jedná o „jednorázové“ připojení (takové připojení, které bude obsahovat jeden požadavek a jednu odpověď), nebo o připojení perzistentního typu, je uložena v hlavičce požadavku a odpovědi. Při implementaci bude nutné pomyslet na to, že navázaná spojení v protokolu HTTP verze minimálně 1.1 jsou implicitně typu perzistentní. V opačném případě, kdy je protokol HTTP verze nižší než 1.1, se nejedná o perzistentní spojení. Navíc bude nutné zohledit informace uložené v HTTP hlavičce, které též mohou definovat typ připojení. [8]

Posledním úkolem třídy *HttpClient* bude doplňovat podepsaná metadata do hlaviček požadavků. Z toho vyplývá, že třída bude muset umět rozeznat bloky *start-line* a *header-field* od *message-body* a doplnit požadovaný obsah do bloku *header-field* (více viz sekce 3.3).



Obrázek 3.1: ISO/OSI model [10]

Vzhledem k případu použití proxy serveru není nutné podporovat všechny typy HTTP požadavků. Nepovinnou součástí implementace budou následující metody:

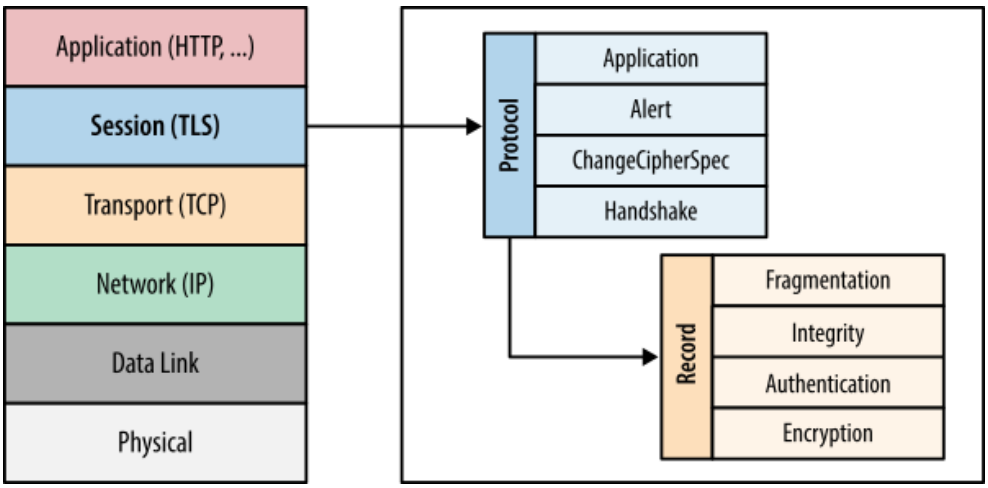
- CONNECT – Slouží k otevření tunelu skrze proxy.
- OPTIONS – Slouží k nastavení komunikace se serverem.
- TRACE – Slouží k trasování cesty k serveru.

Proxy server by měl zaslat klientovi odpověď se statusovým kódem 501 (*Not implemented*) v případě, že obdrží nepodporovaný typ požadavku.

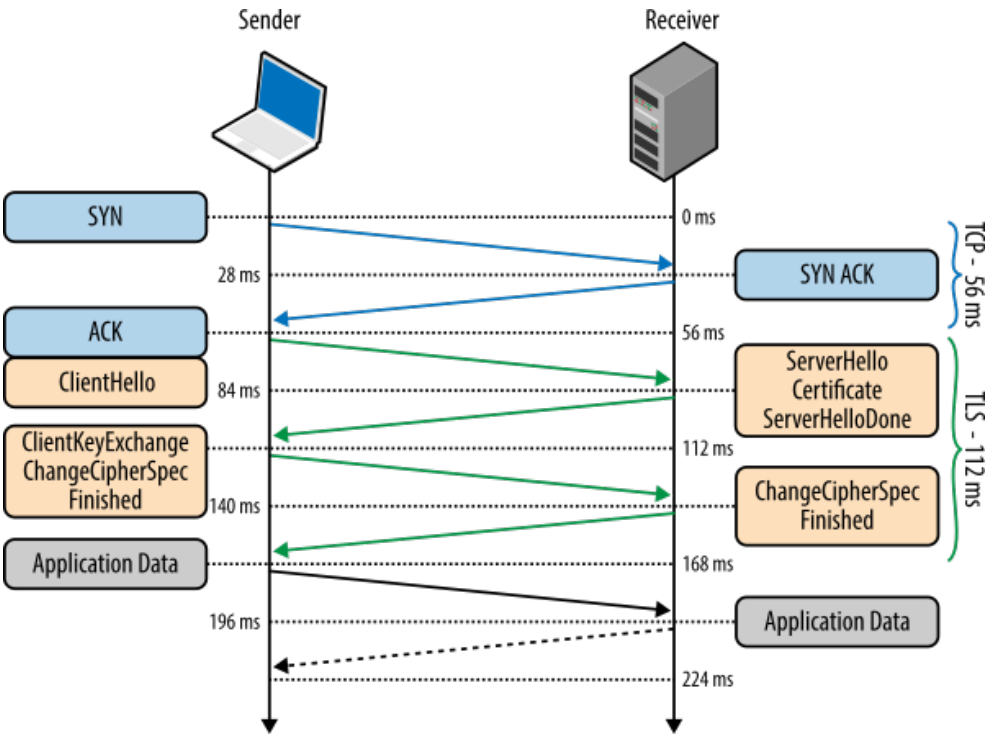
Více informací o podobě protokolu HTTP verze 1.1 se lze dočíst ve standardu RFC 7230 [8].

3.1.3.5 HttpsClient

Tato třída bude odvozená z třídy *HttpClient*. Třída *HttpsClient* bude mít za úkol obstarávat vrsty nad transportní vrstvou v ISO/OSI modelu (viz obr. 3.1) pro protokol HTTPS. Obsah komunikace přicházející do této třídy a odcházející z této třídy bude na rozdíl od protokolu HTTP šifrován. Toto šifrování probíhá na základě použití protokolu TLS. Začlenění TLS protokolu do komunikace znázorňuje obrázek 3.2. Z obrázku vyplývá, že TLS leží v nižší vrstvě než HTTP v rámci ISO/OSI modelu. Veškerá komunikace proto musí být před odesláním zašifrována a naopak při přijetí nejprve dešifrována. Zároveň ihned po otevření spojení se serverem, které bude probíhat prostřednictvím třídy *HttpClient*, bude nutné provést *TLS handshake* (viz obr. 3.3).



Obrázek 3.2: TLS v ISO/OSI modelu [11]



Obrázek 3.3: TLS handshake [12]

V průběhu inicializace *TLS handshake* na straně proxy serveru musí být zajištěno načtení důvěryhodných certifikátů, které budou sloužit jako kotva neboli důvěryhodný zdroj pro ověření certifikačního řetězce, který bude poskytnut cílovým serverem procesu proxy. V případě použití knihovny OpenSSL lze velmi snadno dosáhnout ověření serverového certifikátu [13]. Za předpokladu použití knihovny OpenSSL již bude stačit ověřit pouze návratový kód funkce. I přesto by bylo vhodné doplnit dodatečnou logiku pro vlastní proces ověření certifikátu serveru. Tuto možnost knihovna OpenSSL podporuje.

V případě, že dojde k neúspěšnému *TLS handshake* nebo k chybě v průběhu ověřování certifikátu serveru, je nutné, aby proxy server ukončil spojení, a to jak na straně serveru, tak na straně klienta.

3.1.4 Proxy

Subsekce *Proxy* se věnuje třídám, které budou mít za úkol vytvořit proxy server pro daný protokol (pro protokol HTTP bude výchozí port číslo 80, pro HTTPS 443). Cílem těchto tříd bude naslouchat na portech pro příslušný protokol. Pokud proxy server zachytí příchozí připojení od klienta, dojde k vytvoření nového objektu *Session*, který bude odkazovat na dané příchozí spojení (více informací o objektu *Session* v subsekci 3.1.5). Následně bude tento objekt typu *Session* uložen do fronty.

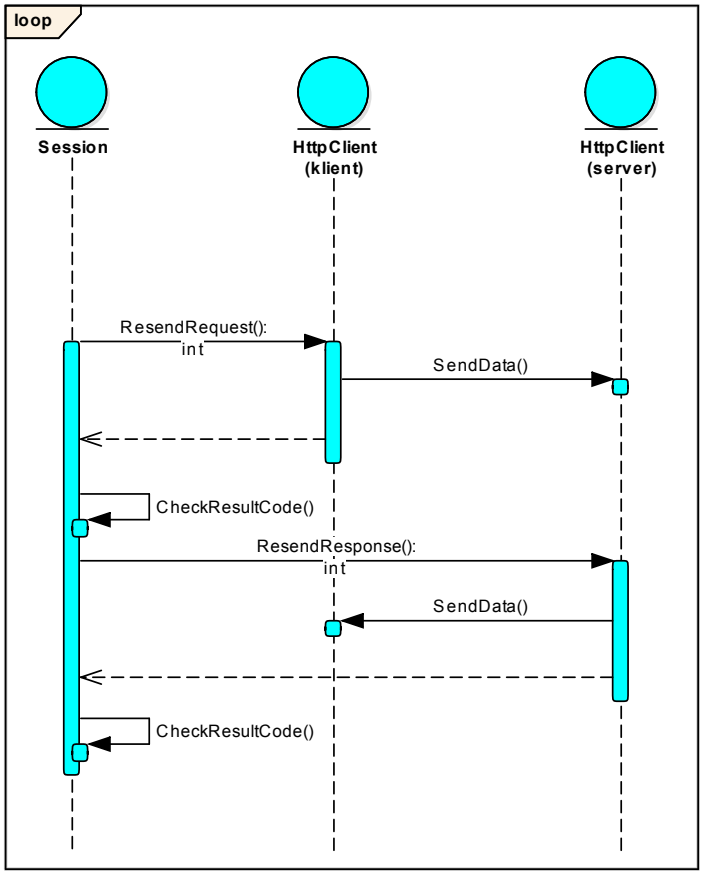
3.1.4.1 ProxyHttps

Třída *ProxyHttps* bude zprostředkovávat proxy pro protokol HTTPS. Součástí této třídy musí být zajištění identifikace proxy serveru certifikátem vůči klientovi. Certifikát použitý pro identifikaci bude zasílán na stranu klienta.

3.1.5 Session

Tato třída pojednává o objektu, který bude sloužit jako obal pro socket definující příchozí připojení klienta na proxy. Součástí položek objektu *Session* bude mimo jiné i typ protokolu, resp. typ proxy, kam se klient připojil, a další potřebné proměnné pro dané protokoly. Funkcí třídy bude správa čili rozdělení přidělených prostředků (viz subsekce 3.1.5.1) mezi spojením s klientem a spojením se serverem. Tyto prostředky budou určeny nastavením proxy serveru. Rozdělení bude řízeno třídami zmíněnými v subsekcích 3.1.3.4 a 3.1.3.5, na základě příchozích HTTP zpráv od koncových subjektů. Časový diagram rozdělení prostředků zachycuje obrázek 3.4.

Objekty *Session* budou používány jako elementy ve frontě. Účelem fronty bude vyčkání na uvolnění prostředků proxy pro příslušný protokol.



Obrázek 3.4: Třída Session – Rozdělení prostředků (schéma)

3.1.5.1 SessionWorker

Třída *SessionWorker* bude sloužit jako jednoduchý „vyřizovač“ požadavků, respektive HTTP zpráv. Jedná se o prostředek proxy serveru a tím pádem může dojít i k jeho vyčerpání v případě velkého počtu příchozích připojení/-požadavků. Účelem této třídy bude vybírat jednotlivé elementy typu *Session* z fronty a poskytování prostředku (sama sebe) pro provedení jejich potřebné komunikace. Tato komunikace se skládá z vyřizování požadavků pocházejících ze strany klienta a odpovědí na tyto požadavky ze vzdáleného serveru.

3.1.5.2 SessionWorkerPool

Třída *SessionWorkerPool* bude mít následující cíle:

- **Správa fronty** – Fronta bude obsahovat elementy *Session* (více viz 3.1.5) a její implementace bude pomocí vzoru producent-konzument. Produ-

centy v tomto případě budou jednotlivé proxy (třída *Proxy*, viz 3.1.4) a konzumenty objekty typu *SessionWorker*.

- **Správa prostředků** – Pro každý proxy server, respektive každý povolený protokol bude vytvořeno n instancí třídy *SessionWorker*.

3.1.6 Sockety

Subsekce socketů se bude zabývat základní správou socketů. Správa socketů může být odlišná na základě platformy (minimálně pro operační systémy Unix-like a Windows). Z tohoto důvodu by součástí implementace mělo být API (abstraktní předek), které bude zastřešovat volání správy socketů a dosáhne se tak jednotného multiplatformního použití. Rozhraní třídy bude obsahovat minimálně následující metody:

- **Connect** – Vytvoření socketu a připojení k cíli.
- **Close** – Uzavření socketu.
- **Bind** – Namapování portu na socket.
- **Listen** – Zapnutí naslouchání na daném socketu.
- **Accept** – Zpracování příchozího připojení, vytvoření nového socketu.

Volání tříd spadajících do této subsekce by mělo probíhat výhradně z tříd uvedených v subsekcích 3.1.3.4, 3.1.3.5 a 3.1.4. Zmíněné třídy souvisí se správou síťových připojení čili připojením klientů nebo připojením ke vzdálenému serveru.

3.1.7 Správa

Subsekce *Správa* obsahuje návrh tříd, které nebudou zařazeny do žádné z předchozích subsekcí. Bude se jednat o třídy, které budou mít za úkol:

- správu inicializace aplikace proxy serveru (inicializace konfigurace, logování, proxy, ...) – více v sekci 3.2,
- ukončení procesu proxy serveru (vypnutí jednotlivých proxy, uvolnění paměti, ...).

3.2 Inicializace

Kroky inicializace, které bude nutné provést před spuštěním přenosu dat přes proxy, budou probíhat v následujícím pořadí:

1. inicializace dat pomocí externího procesu Wrapper,
2. inicializace samotného procesu proxy.

3.2.1 Inicializace dat

Inicializace dat bude probíhat přes Wrapper, který poběží nezávisle na proxy. Účelem tohoto procesu bude správa dat potřebných pro inicializaci proxy serveru. Do těchto dat spadají: certifikáty, privátní klíče, sdílená tajemství. Proces bude společně s aplikací proxy startovat při bootu systému. Součástí správy těchto dat jsou akce stažení a smazání inicializačních dat.

Proces aplikace Wrapper, který bude sloužit k inicializaci dat aplikace proxy serveru, je logicky oddělenou částí od proxy serveru a proto není součástí této práce.

3.2.1.1 Stažení inicializačních dat

Inicializační data budou stahována ze vzdáleného serveru. Aby nedošlo ke stažení inicializačních dat třetí stranou a jejich následnému zneužití, je nutné provést autentizaci zdroje, který zasílá požadavek o inicializační data. Data budou následně uložena na disk v podobě souborů. Proces autentizace je popsán v subsekci 3.2.3.

3.2.1.2 Smazání inicializačních dat

Ke smazání inicializačních dat bude docházet až ve chvíli, kdy budou všechny inicializační data úspěšně načtena v paměti procesu proxy. V takový okamžik již nebude potřeba, aby byla součástí souborového systému. Jejich smazání by ideálně mělo proběhnout pomocí přepsání náhodnými nebo nulovými bajty.

3.2.2 Inicializace proxy

Před spuštěním přenosu dat v procesu proxy serveru bude předcházet:

1. nastavení konfigurace,
2. inicializace logování,
3. inicializace kryptografické knihovny OpenSSL,
4. načtení inicializačních dat,
5. inicializace proxy serverů (jednotlivé protokoly),
6. jiné části subsystému aplikace proxy.

Pokud tato část inicializace proběhne bez chyb, budou spuštěny proxy servery pro povolené protokoly na základě svých konfigurací. O získání inicializačních dat se zmiňuje subsekce 3.2.1.

3.2.3 Autentizace

Proces autentizace bude probíhat jako součást externího procesu Wrapper. Autentizace externího procesu bude prováděna za účelem prokázání totožnosti, aby nemohlo dojít ke stažení inicializačních dat ze vzdáleného serveru třetí stranou.

K autentizaci bude použit privátní klíč (asymetrická kryptografie). Tento privátní klíč nebude uložen na lokálním počítači. K jeho načtení dojde pomocí externího úložiště, např. USB klíče. Hlavním důvodem použití externího úložiště je bezpečnost, která je popsána sekci 3.4.

3.3 Identifikace proxy

Pro zjištění původu dat je nutné přidat do HTTP zpráv informaci o zdroji. Tyto informace budou sloužit k identifikaci prostředí, ve kterém byl proces proxy serveru spuštěn. Aby mohl cílový server nakládat s těmito informacemi jako důvěryhodnými, bude nutné zajistit, aby jejich součástí byl i digitální podpis nebo jiný objekt, který bude zaručovat autentifikaci a integritu dat obsahujících tyto informace.

Objekt zajišťující výše uvedené požadavky bude založen na algoritmu HMAC. Tento algoritmus je sám o sobě stále považován za bezpečný čili neprolomený. HMAC je založen na sdíleném tajemství a zvolené hash funkci (navrhovanou hash funkcí v této práci je funkce sha256). Autentifikace je založena na společném tajemství, zatímco integrita dat je dosažena použitím neprolomené hash funkce[4]. [14]

3.3.1 Vložení identifikačních dat

Do požadavků HTTP budou doplněny následující konkrétní položky obsahující identifikační data (založeno na analýze 2.4):

1. časové razítko (unix timestamp),
2. identifikátor učebny (založen na IP adrese počítače),
3. řetězec v hexadecimálním kódování obsahující výsledný HMAC-SHA256.

Vstupní data do algoritmu HMAC budou obsahovat pouze položky, které budou doplněné do HTTP zprávy (viz výčet výše). Tyto položky budou před výpočtem HMAC serializovány tak, že se použije jejich název a hodnota.

[4]Vzhledem k faktu existence společného tajemství a za předpokladu, že útočník nezná toto tajemství, by bylo možné použít slabší nebo případně i prolomenou hash funkci.

3.3.2 Ověření identifikačních dat

Na straně vzdáleného serveru bude docházet ke kontrole obsahu HTTP hlaviček, které jsou součástí přicházejících požadavků z klienta (lokálního počítače v učebně FIT ČVUT). Server bude znát společné tajemství sdílené s klientem. Ověření příchozích požadavků bude probíhát v následujících krocích:

1. **Ověření obsahu HTTP hlavičky** – Kontrola existence všech požadovaných položek.
2. **Kontrola časového razítka** – Časový rozdíl mezi aktuálním časem a časem uvedeným v razítku. Pro tento krok bude nutné zvolit vhodný interval tolerance a zajistit dostatečnou synchronizaci času klienta a vzdáleného serveru. V případě, že rozdíl časových údajů bude spadat mimo interval tolerance, dojde k zamítnutí požadavku. Tato kontrola slouží k tomu, aby požadavek odposlechnutý útočníkem nemohl být v budoucnu znovu použit, tj. opět poslán na server a úspěšně ověřen stranou serveru.
3. **Kontrola podpisu** – Výpočet HMAC-SHA256 a porovnání s příchozí hodnotou v HTTP hlavičce. Pokud se hodnoty nebudou shodovat, bude požadavek zamítnut.

Příchozí požadavek bude považován za důvěryhodný, pokud projde všemi výše uvedenými kontrolami.

3.4 Bezpečnost

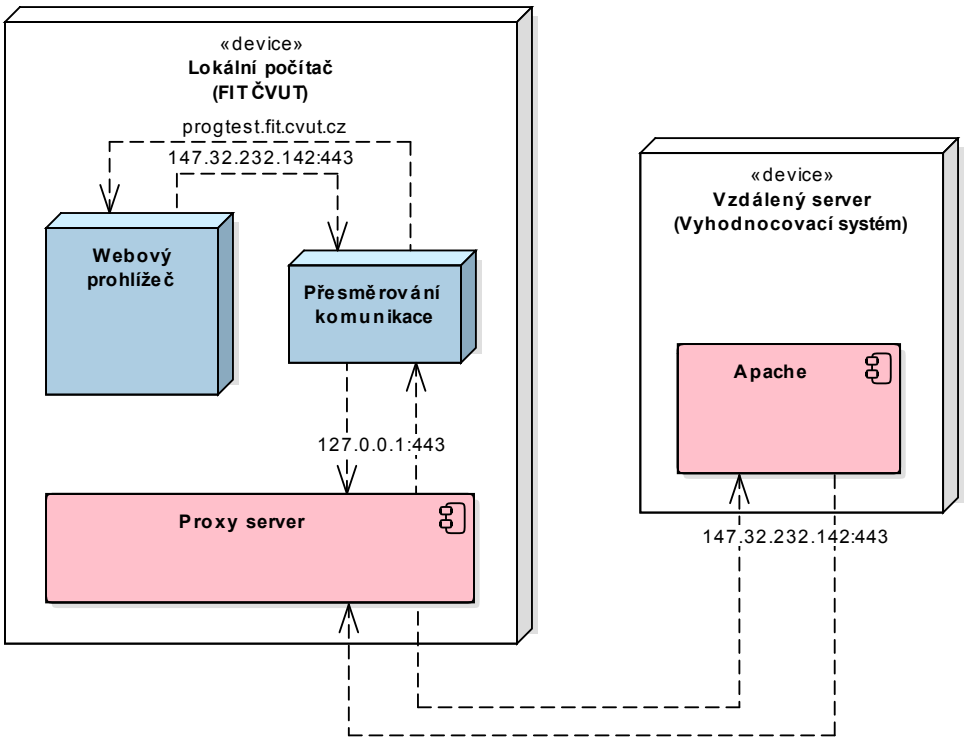
Tato sekce popisuje jakým způsobem budou řešeny jednotlivé faktory rizik, které jsou součástí sekce 2.5.

3.4.1 Souborový systém

Zabezpečení dat uložených v souborovém systému testovacího prostředí *ProgtestEnv* je založeno zejména na uživatelských oprávněních a minimalizaci délky intervalu uložených souborů na disku. Uživatelská oprávnění do příslušných složek obsahující citlivá data budou nastavena tak, aby se k jejím obsahům nedostal běžný uživatel (student). Zároveň budou neprodleně mazány, resp. přepisovány ve chvíli, kdy už nebudou potřeba.

3.4.2 Komunikace

Komunikace mezi proxy a serverem bude probíhat ve stejném protokolu, jako bylo navázáno spojení mezi klientem a proxy, čili protokol HTTPS. Bezpečnost protokolu HTTPS je opřena v dnešní době výhradně o protokol TLS [15].



Obrázek 3.5: Návrh modelu nasazení

Zatímco v komunikaci mezi klientem a proxy se bude používat certifikát vygenerovaný vlastní certifikační autoritou (konkrétněji vlastním kořenovým certifikátem), tak v komunikaci mezi proxy a serverem bude použit certifikát koncového serveru vydaný plnohodnotnou certifikační autoritou.

3.4.3 Privátní klíč pro autentizaci

Zabezpečení uložení a distribuce privátního klíče pro autentizaci aplikace Wrapper je postaveno zejména na lidském faktoru a striktních pravidlech jeho distribuce. Distribuce na lokální počítače probíhá pouze při inicializaci operačního systému a v průběhu tohoto procesu by se v místnosti neměl nacházet žádný student.

3.5 Nasazení

Výsledná aplikace proxy serveru bude nasazena do testovacího prostředí *ProgtestEnv* společně s externí aplikací Wrapper zajišťující autentifikaci a správu inicializačních dat. V nastavení prostředí bude nutné zajistit přesměrování potřeb-

ných protokolů/domén na vstupní porty lokálního proxy serveru. Diagram nasazení zachycuje obrázek 3.5.

3.6 Použité nástroje a technologie

V dnešní době je na výběr mnoho cest, jakým způsobem lze implementovat proxy server. Z pohledu programovacích jazyků se jedná jak o jazyky, které jsou více nízkoúrovňové (např. programovací jazyk C), tak i vysokoúrovňové (např. Java). Vzhledem k typu problematiky této práce, požadovanému prostředí (operačnímu systému distribuce Linux) a s přihlédnutím k vlastním zkušenostem jsem vybral následující nástroje a technologie pro vytvoření požadované výsledné aplikace. Jednotlivé body mohou obsahovat i odůvodnění volby.

- Programovací jazyk C++ (standard C++11)
  - Možnost protěžení výkonu.
  - Možnost pracovat i více nízkoúrovňově (jazyk C).
  - Dlouhodobá osobní zkušenost a dobrá znalost.
- Kryptografická knihovna OpenSSL[5]
  - Robustní a ověřené multiplatformní řešení.
  - Podpora všech potřebných kryptografických prvků a funkcí (objekty, algoritmy, protokoly, generování certifikátu ...).
  - Osobní zkušenost.
- Vývojové prostředí Microsoft Visual Studio (Windows)
- Kompilační nástroj Makefile (Unix)

[5] https://www.openssl.org/.

Kapitola 4

Implementace a realizace

Obsahem této kapitoly je zejména náhled na provedení praktické části, která je založena na předcházející kapitole obsahující návrh podoby a struktury aplikace (viz kapitola 3).

4.1 Načtení vstupních dat

Načtení vstupních dat probíhá pouze v inicializační části procesu proxy. Načítaná data se dělí do dvou hlavních skupin:

1. konfigurace,
2. kryptografické objekty.

V prvním případě (konfigurace) se stará o načtení dat třída *Config* (více viz subsekce 3.1.2.2) a jedná se pouze o explicitní načtení konfiguračního souboru. V opačných případech, kdy se nejedná o konfigurační soubor, se všechna vstupní data načítají do třídy *DataMgr*. Třída *DataMgr* slouží jako úložiště všech potřebných souborů neboli dat pro správný běh aplikace proxy serveru v protokolu HTTPS.

Pokud se nepodaří načíst konfigurační soubor, v jeho obsahu se nedohledá požadovaná proměnná nebo načtená proměnná obsahuje neplatnou hodnotu, je automaticky použita výchozí hodnota proměnné, která je definována přímo ve zdrojovém kódu třídy *Config*. Jméno konfiguračního souboru je pevně stanoveno na hodnotu "radford.cfg". Formát tohoto souboru je popsán v subsekci 4.1.1.

4.1.1 Formát vstupních dat

Tato subsekce pojednává o použitém formátu konfiguračního souboru, sloužícího pro inicializaci aplikace proxy serveru, a vyžadovaném vstupním formátu kryptografických objektů.

```
#
# Povolene verze internetoveho protokolu.
# Bitove priznaky: 1 - IPv4 [default]
#                  2 - IPv6
#                  3 - IPv4 a~IPv6
#
Net.Protocols.Enabled=1
#
########################################
# SSL
########################################
#
# Cesta ke koncovemu certifikatu v~PEM formatu - Serverovy certifikat
    proxy
#
SSL.Proxy.EndCert.Path=end.cert.pem
#
```

Ukázka 4.1: Výřez z konfiguračního souboru radford.cfg

4.1.1.1 Konfigurační soubor

Syntaxi řádku konfiguračního souboru lze definovat následujícím regulárním výrazem:

```
^((#.*)|((([a-zA-Z]+(\.)?))+=(.*)))$
```

Řádky začínají znakem "#"jsou vynechány z načítání dat. Takové řádky jsou určeny pouze jako doplňkové, mohou nést užitečnou informaci pro uživatele, který tento konfigurační soubor upravuje. Část vyplněného konfiguračního souboru lze vidět na ukázce 4.1.

4.1.1.2 Kryptografické objekty

Mezi načítané kryptografické objekty se řadí certifikáty, privátní klíče a sdílená tajemství. Interní formát certifikátů v aplikaci je založen na struktuře X.509 z knihovny OpenSSL. Interní formát privátních klíčů je založen na struktuře EVP_PKEY (taktéž knihovna OpenSSL). Obě tyto struktury podporují mimo jiné formát PEM. Načtení těchto struktur je implementováno pouze z PEM formátu [16]. Data do obou objektů je možné načíst jak ze souboru, tak z textového pole (*buffer*). Sdílené tajemství je načteno ze souboru jako textový řetězec.

Předání dat u certifikátů a privátních klíčů probíhá přes objekt *BIO*. „*A BIO is an I/O abstraction, it hides many of the underlying I/O details from*

an application. If an application uses a BIO for its I/O it can transparently handle SSL connections, unencrypted network connections and file I/O.“ [17]

4.2 Konfigurace

O konfiguraci aplikace se stará třída *Config*, jejíž návrh je součástí subsekce 3.1.2.2. Obsahem této sekce je vyzdvihnutí důležitých nastavení, která lze konfigurovat.

- **Zapnutí protokolu** – Umožňuje zapnout/vypnout každý protokol zvlášť.
- **Nastavení portu** – Umožňuje nastavit vstupní port, na kterém má proxy naslouchat pro daný protokol.
- **Verze internetového protokolu** – Umožňuje vybrat podporu IPv4 nebo IPv6[6].
- **Důvěryhodné certifikáty (HTTPS)** – Umožňuje nastavit cestu k vlastním důvěryhodným certifikátům nebo použití systémového úložiště.
- **Vlastní ověření certifikátu (HTTPS)** – Umožňuje zvolit vlastní algoritmus pro ověření certifikátu serveru.
- **Výběr způsobu identifikace** – Umožňuje zvolit modul, který bude použit pro vytvoření identifikace v HTTP zprávách.

Zbylá vstupní nastavení, která je možné provést a nejsou zde zároveň uvedena, lze najít v souboru "radford.cfg", který je součástí obsahu přiloženého CD.

4.3 Vytvoření spojení a správa session

Pro všechna přijatá spojení jsou vytvořeny objekty třídy *Session* a tyto objekty jsou následně zařazeny do fronty, která je asynchronně zpracovávána tzv. *workery* (viz 3.1.5.1). Zdrojový kód implementující přijetí spojení a zařazení do fronty pro protokol HTTP je uveden v ukázce 4.2.

Workflow diagram přijetí spojení, zařazení do fronty a následné zpracování (vyřízení požadavků) je zachycen na obrázku C.1, který je součástí přílohy. Objekt *Session* zaniká ve chvíli, kdy dojde k uzavření spojení na obou stranách, tj. s klientem a serverem. O jeho uvolnění se stará přímo *worker* (viz ukázka D.1).

[6] IP verze 4 a 6.

```
void ProxyHttp::Accept(SessionWorkerPool& pool)
{
   LOG_TRACE("ProxyHttp::Accept");

   int iClientSocketFD = _pSocketConn->Accept(_iSocketFD);

   if (iClientSocketFD <= 0)
   {
      LOG_ERROR("ProxyHttp::Accept - Error while accepting client");
      return;
   }

   LOG_DEBUGF("ProxyHttp::Accept - Client connected [Socket: %i]",
       iClientSocketFD);

   pool.EnqueueSession(new Session(iClientSocketFD, _info));
}
```

Ukázka 4.2: Zdrojový kód funkce ProxyHttp::Accept

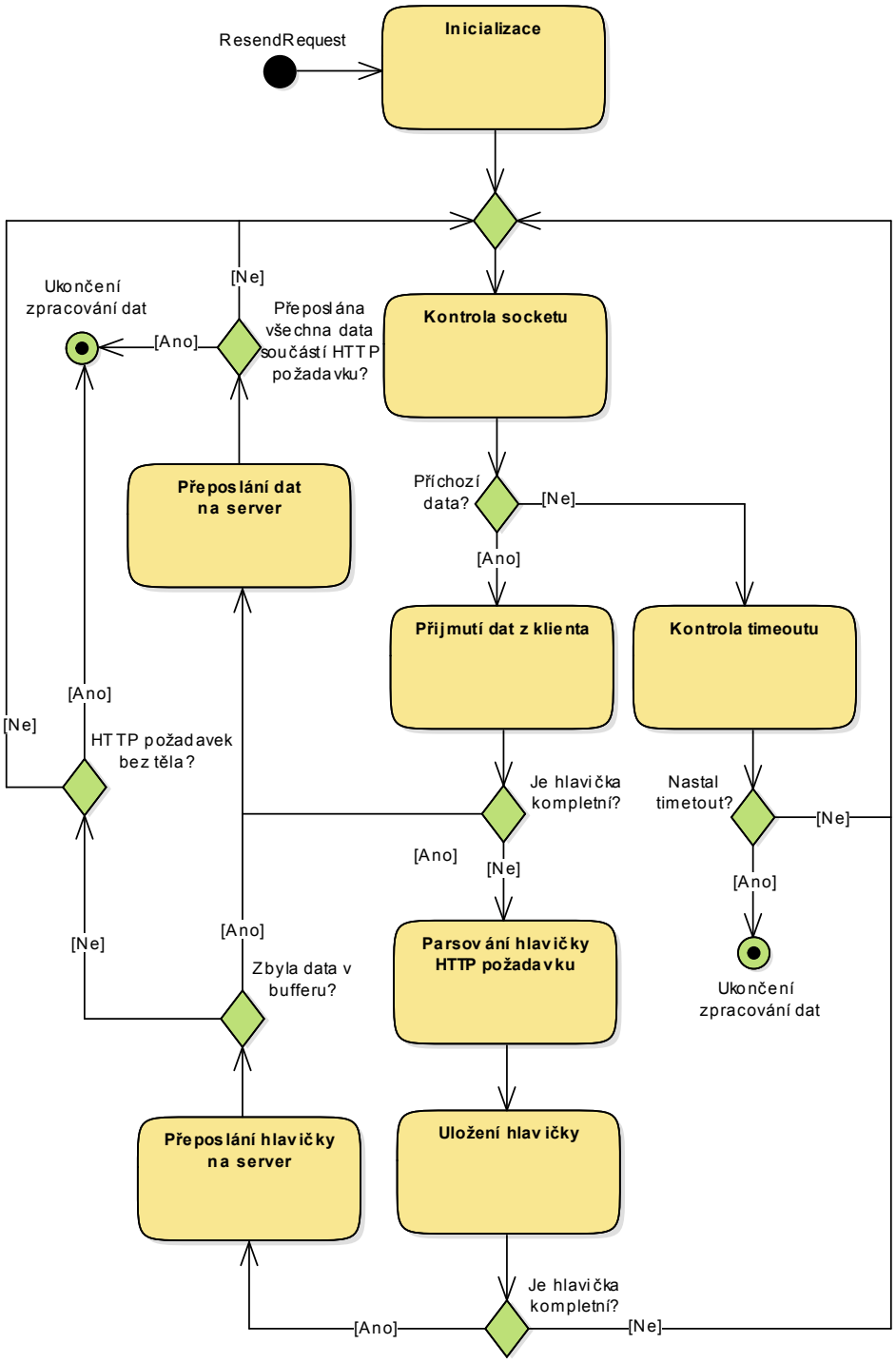
4.4 Zpracování a přeposlání příchozích dat

Data do proxy přicházejí ze dvou zdrojů, jimiž jsou klient a server. Vzhledem k faktu, že oba zdroje odesílají HTTP zprávy, je zpracování zpráv z obou zdrojů velmi podobné. O získání, zpracování a přeposlání příchozích dat druhému subjektu se starají funkce *ResendRequest* a *ResendResponse*. Funkce *ResendRequest* je volána na příchozí data z klienta (požadavky), zatímco *ResendResponse* slouží naopak pro data ze serveru (odpovědi). Obě funkce jsou součástí třídy *HttpClient* (viz subsekce 3.1.3.4). Chování těchto funkcí je znázorněno diagramem 4.1.

V případě protokolu HTTPS je navíc nutné příchozí data dešifrovat pomocí veřejného klíče koncového certifikátu serveru a odchozí naopak zašifrovat privátním klíčem koncového certifikátu proxy.

4.5 Nastavení certifikátů proxy

Nastavení certifikátů proxy se týká pouze protokolu HTTPS. Toto nastavení probíhá ve funkci *Open*, která náleží třídě *ProxyHttps*. K volání této funkce dochází ihned po vytvoření naslouchání procesu na daném portu čili vytvoření socketu. Vytvoření naslouchání zajišťuje posloupnost funkcí *Bind* a *Listen*. Nastavení protokolu TLS (kryptografická složka protokolu HTTPS) se ukládá do tzv. SSL kontextu (objekt SSL_CTX). Tento kontext je nutné nejprve vytvořit, v případě této práce se jedná o funkci *SSL_CTX_new* [18]. Součástí této funkce je též povinný parametr určující typ kryptografického protokolu,



Obrázek 4.1: Třída HttpClient – Diagram funkce ResendRequest

na kterém má být kontext vytvořen (vybraným protokolem v implementaci je protokol TLS verze 1.3). SSL kontext je používán knihovnou OpenSSL a slouží k nastavení protokolu SSL/TLS. Taktéž umožňuje zakázat určité šifry nebo například starší protokoly, které nemají být použity pro komunikaci.

Nezbytnou součástí pro úspěšné a zároveň správné nastavení certifikátů do SSL kontextu jsou následující data:

1. koncový certifikát, kterým se bude proxy prokazovat,
2. privátní klíč, který patří do páru s veřejným klíčem koncového certifikátu,
3. mezilehlé certifikáty ležící v certifikačním řetězci (volitelné na základě délky certifikačního řetězce a důvěryhodných certifikátů na straně klienta),

4.6 Vlastní ověření certifikátu vzdáleného serveru

Jakým způsobem lze postupovat při ověření certifikátu pomocí interního systému knihovny SSL je popsáno v subsekci 3.1.3.5. V případě volby použití vlastního algoritmu pro ověření certifikátu se nabízí dvě následující cesty:

- nastavení callback funkce pomocí *SSL_CTX_set_cert_verify_callback*,
- po handshake zavolat vlastní implementaci (do)ověření.

V implementaci této práce je zvolena druhá možnost. Vlastní ověření certifikátu se skládá z následujících bloků:

1. **Ověření koncového certifikátu** – Kontrola validity a platnosti podpisu.
2. **Ověření certifikačního řetězce** – Bez kontroly revokace, pomocí funkce *X509_verify_cert*.

Ověření certifikačního řetězce probíhá nad certifikáty, které byly obdrženy při *TLS handshake* se vzdáleným serverem. Získané certifikáty lze získat opět z SSL kontextu.

4.7 Zavedení do startu systému

Aplikace proxy serveru musí být spuštěna v průběhu startu systému. Aby došlo ke spuštění procesu, je nutné nahrát skript do příslušné složky v souborovém systému a následně použít příkaz pro zavedení skriptu do bootu systému (může se lišit dle linuxové distribuce). Podrobný popis postupu, jak docílit přidání spuštění aplikace proxy serveru do bootu systému, lze nalézt na přiloženém CD.

4.7.1 Oprávnění

Je nutné zajistit, aby uživatel, který spouští daný skript (obsahující mimo jiné i spuštění aplikace proxy serveru), měl dostatečná oprávnění pro vykonání všech příkazů, která jsou součástí onoho skriptu. Potřebná oprávnění jsou taktéž nutná zajistit pro samotnou inicializaci procesu proxy. Uživatelská oprávnění v inicializaci proxy serveru jsou vyžadována v následujících operacích:

- čtení konfiguračního soboru a inicializačních souborů,
- zapisování do logovacího souboru (pokud je zvoleno),
- otevření lokálního portu (zejména v případě, kdy je zvolena hodnota portu menší než 1000).

4.8 Vystavení certifikátu pro proxy

Aby proxy server pro protokol HTTPS mohl správně fungovat, je nezbytnou podmínkou vydat koncový certifikát, kterým se daný proxy server bude prokazovat. Tímto vydaným certifikátem bude proxy server prokazovat svoji totožnost vůči klientovi v průběhu *TLS handshake*.

Na podobu vystavěného certifikátu jsou kladeny určité podmínky/pravidla, které je nutné dodržet, jinak nebude certifikát vyhodnocen jako platný nebo důvěryhodný. Jedná se o následující omezení:

- Certifikát, kterým se proxy prokazuje, nesmí být *self-signed* (sám sebou podepsaný).
- V CN (*CommonName*) musí být uvedena doména minimálně druhého řádu, která je shodná s doménou, na které je certifikát použit. Též lze využít tzv. *Wildcard certifikát*[7].
- Musí se jednat o serverový certifikát.
- Nesmí se jednat o certifikát certifikační autority.
- Klíč certifikátu musí sloužit k digitálnímu podpisu, šifrování a autentifikaci serveru.

Obsah této sekce byl čerpán ze zdrojů [19] a [20].

[7] http://wiki.cacert.org/WildcardCertificates.

4.8.1 Vystavení certifikátu pomocí knihovny OpenSSL

Tato subsekce popisuje, jakým způsobem byly v této práci vystaveny certifikáty pro proxy server. Celý postup včetně všech potřebných skriptů a konfiguračních souborů (pro vydání všech nezbytných certifikátů) lze nalézt na přiloženém CD.

Pro vydání koncového certifikátu serveru je nutné nejdříve vydat certifikát kořenový (tzv. *root*). Vzhledem k faktu, že kořenový certifikát je *selfsigned*, stačí pouze vygenerovat privátní klíč, který bude používán daným kořenovým certifikátem. Následně je nutné příslušně zavolat knihovnu OpenSSL pro vystavení zmiňovaného kořenového certifikátu. V průběhu generování kořenového certifikátu bude vyžadováno heslo k privátnímu klíči. Tento privátní klíč bude použit pro podpis certifikátu.

Vydání koncového/mezilehlého certifikátu (pro proxy server) je výrazně složitější proces než u certifikátu kořenového. Celý proces probíhá v následujících krocích:

1. **Vygenerování privátního klíče** koncového/mezilehlého **certifikátu**.
2. **Vytvoření žádosti o** koncový/mezilehlý **certifikát** – Žádost musí splňovat požadavky uvedené v sekci 4.8.
3. **Podepsání** vytvořené **žádosti privátním klíčem certifikátu** z předchozích kroků.
4. **Vygenerování certifikátu na základě podepsané žádosti**.
5. **Podepsání** koncového/mezilehlého **certifikátu privátním klíčem vydavatele certifikátu**.

Vygenerované certifikáty lze uložit například do formátu PEM (textový formát – viz ukázka D.2) nebo DER (binární formát). Tyto formáty jsou ekvivalentní po stránce obsahu (jedná se pouze o jiné kódování) a lze je vzájemně převádět. Příklad informací získaných z vystaveného koncového certifikátu v PEM formátu určeného pro identifikaci proxy serveru je součástí ukázky 4.3. Součástí této ukázky není modulo veřejného klíče certifikátu a hodnota podpisu certifikátu.

Obsah subsekce byl čerpán ze zdroje [21].

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 4104 (0x1008)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=CZ, ST=Czech Republic, O=FIT CVUT, CN=Root Test
        Validity
            Not Before: Apr 16 17:44:06 2017 GMT
            Not After : Apr 16 17:44:06 2018 GMT
        Subject: C=CZ, ST=Czech Republic, O=FIT CVUT,
            CN=progtest.fit.cvut.cz
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Cert Type:
                SSL Server
            Netscape Comment:
                OpenSSL Generated Server Certificate
            X509v3 Subject Key Identifier:
                4F:72:29:AA:C4:1C:3A:C0:DD:2F:43:4C:43:D0:B4:85:27:6E:F8:5E
            X509v3 Authority Key Identifier:
                keyid:20:32:80:F0:51:19:55:04:B2:22:3E:DF:A9:EF:71:65:26:CD:3D:6E
                DirName:/C=CZ/ST=Czech Republic/O=FIT CVUT/CN=Root Test
                serial:C8:7F:35:B5:F2:29:1C:FF

            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
    Signature Algorithm: sha256WithRSAEncryption
```

Ukázka 4.3: Informace o koncovém certifikátu pro proxy

Kapitola 5

Testování

Testování aplikace proxy serveru probíhalo na operačním systému distribuce Linux. Součástí testování bylo zavedení aplikace do startu systému, viz sekce 4.7. Všechna potřebná inicializační data byla již součástí souborového systému (simulace použití externí aplikace Wrapper), aby došlo k úspěšné inicializaci aplikace proxy serveru. Vybraným webovým prohlížečem pro testování proxy serveru byl zvolen prohlížeč Mozilla Firefox verze 45.2.0.

5.1 Vstupní nastavení

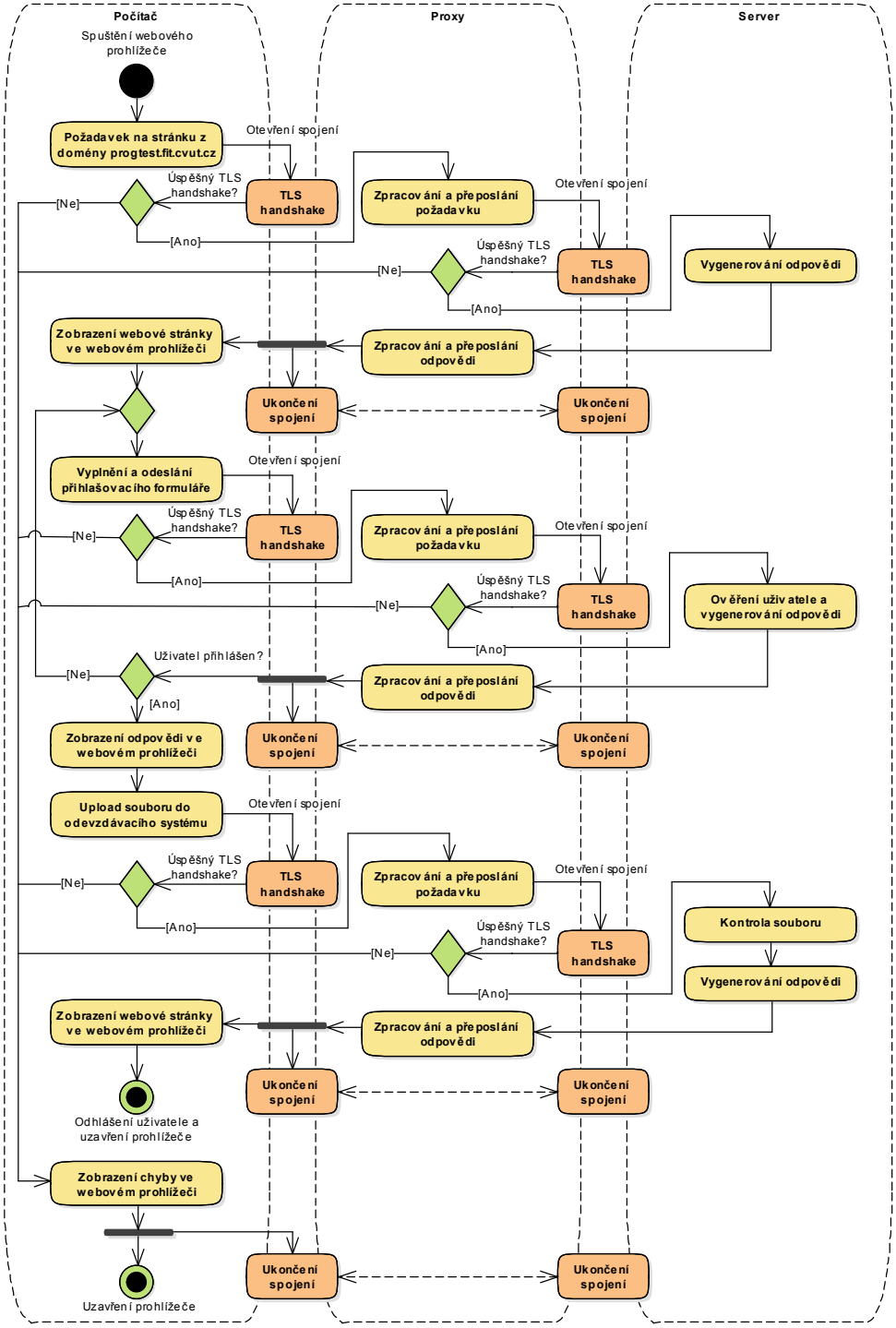
Tato sekce popisuje nejdůležitější body vstupního nastavení proxy serveru, která byla použita v případě všech testovacích scénářů.

- Povolené protokoly: HTTPS
- Povolené domény: progtest.fit.cvut.cz
- Vynucení identifikace HTTP zpráv: Ano
- Použití systémového úložiště důvěryhodných certifikátů: Ano
- Způsob ověření koncového certifikátu serveru: Implicitní volání kryptografické knihovny

5.2 Testovací scénáře

5.2.1 Pozitivní scénář (protokol HTTPS, povolený protokol a doména)

Tento scénář je jediným pozitivním scénářem použití aplikace proxy serveru. Součástí scénáře je běžný pohyb uživatele na webových stránkách domény progtest.fit.cvut.cz. Průběh scénáře je zachycen diagramem 5.1.



Obrázek 5.1: Pozitivní testovací scénář

5.2.2 Negativní scénář (protokol HTTPS, nedůvěryhodný certifikát proxy)

Tento scénář má skončit neúspěšným načtením požadované webové stránky na doméně progtest.fit.cvut.cz z důvodu neúspěšného *TLS handshake* mezi webovým prohlížečem a proxy, konkrétně kvůli neúspěšnému ověření koncového certifikátu proxy. V tomto testovacím scénáři schází v použitém webovém prohlížeči kořenový certifikát, kterým je podepsán koncový certifikát proxy serveru.

Průběh scénáře v bodech:

1. spuštění webového prohlížeče,
2. otevření webové stránky na doméně progtest.fit.cvut.cz,
3. výskyt chyby při otevření webové stránky,
4. uzavření webového prohlížeče.

Chybový stav, který nastal v průběhu pokusu o načtení webové stránky: *Your connection is not secure (progtest.fit.cvut.cz uses an invalid security certificate. The certificate is not trusted because the issuer certificate is unknown.)*. Tento stav odpovídá očekávanému stavu.

5.2.3 Negativní scénář (protokol HTTPS, nedůvěryhodný certifikát serveru)

Tento scénář má skončit neúspěšným načtením požadované webové stránky na doméně progtest.fit.cvut.cz z důvodu neúspěšného *TLS handshake* mezi proxy a serverem. V tomto testovacím scénáři je změněno vstupní nastavení úložiště důvěryhodných certifikátů – není použito žádné úložiště důvěryhodných certifikátů čili schází důvěryhodný certifikát k doméně progtest.fit.cvut.cz.

Průběh scénáře v bodech:

1. spuštění webového prohlížeče,
2. otevření webové stránky na doméně progtest.fit.cvut.cz,
3. výskyt chyby při otevření webové stránky,
4. uzavření webového prohlížeče.

Chybový stav, který nastal v průběhu pokusu o načtení webové stránky: *Secure Connection Failed (The connection to the server was reset while the page was loading.)*. Tento stav odpovídá očekávanému stavu.

5.2.4 Negativní scénář (protokol HTTPS, nepovolená doména)

Tento scénář má skončit neúspěšným načtením požadované webové stránky na doméně edux.cvut.cz z důvodu pokusu o přístup na webovou stránku, která není součástí množiny povolených domén v proxy serveru.

Průběh scénáře v bodech:

1. spuštění webového prohlížeče,
2. otevření webové stránky na doméně edux.fit.cvut.cz,
3. výskyt chyby při otevření webové stránky,
4. uzavření webového prohlížeče.

Chybový stav, který nastal v průběhu pokusu o načtení webové stránky: *Secure Connection Failed (The connection to the server was reset while the page was loading.)*. Tento stav odpovídá očekávanému stavu.

5.2.5 Negativní scénář (protokol HTTP, nepovolený protokol)

Tento scénář má skončit neúspěšným načtením požadované webové stránky na doméně fit.cvut.cz z důvodu pokusu o přístup na webovou stránku přes protokol HTTP, který není podporovaný proxy serverem.

Průběh scénáře v bodech:

1. spuštění webového prohlížeče,
2. otevření webové stránky na doméně fit.cvut.cz,
3. výskyt chyby při otevření webové stránky,
4. uzavření webového prohlížeče.

Chybový stav, který nastal v průběhu pokusu o načtení webové stránky: *The connection was reset (The connection to the server was reset while the page was loading.)*. Tento stav odpovídá očekávanému stavu.

Závěr

Cílém mé bakalářské práce bylo analyzovat stávající řešení zabezpečení komunikace mezi učebnovými počítači a vzdáleným vyhodnocovacím systémem postavené na nástroji stunnel. Následně provést návrh a implementaci vlastního vylepšeného řešení na myšlence ssl-to-ssl proxy.

V rámci implementace se vyskytl značný problém s použitím kryptografického čipu TPM, který měl být původně součástí procesu autentizace systému. Tento problém byl nakonec vyřešen nalezením jiného a zároveň lepšího východiska, které bude sloužit pro autentizaci.

Výsledná aplikace proxy serveru byla navržena s cílem jednoduchého budoucího doplnění o další logické části v případě potřeby. Z pohledu implementace byly splněny všechny požadavky na funkcionalitu. Pro návrh vnitřní struktury aplikace byly použity osvědčené návrhové vzory.

K vypracování práce byly mimo jiné použity i standardy RFC, o které se zejména opírá značná část návrhu a implementace aplikace související se zpracováním protokolů.

Pro produkční nasazení aplikace proxy serveru do testovacího prostředí *ProgtestEnv* je nutné doimplementovat externí aplikaci (Wrapper), která se bude starat o správu inicializačních dat a autentizaci operačního systému. Aplikace Wrapper není součástí této bakalářské práce, jedná se o logicky oddělenou problematiku. Bakalářská práce obsahuje pouze návrh, jakým způsobem by tato externí aplikace mohla pracovat.

Aplikace proxy serveru je připravena pro integraci do další verze prostředí *ProgtestEnv* a lze ji aktuálně použít za předpokladu ručního nastavení potřebných inicializačních dat.

Literatura

[1] Peterka, J.: *Báječný svět elektronického podpisu.* CZ.NIC, z. s. p. o., Duben 2011, ISBN 978-80-904248-3-8, [cit. 2017-05-09].

[2] Lloyd, S.: Understanding Certification Path Construction [online]. Září 2002, [cit. 2017-05-04]. Dostupné z: `http://www.oasis-pki.org/pdfs/Understanding_Path_construction-DS2.pdf`

[3] D. Cooper, S. F., S. Santesson: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile [online]. Technická zpráva, Květen 2008, [cit. 2017-05-04]. Dostupné z: `https://www.ietf.org/rfc/rfc5280.txt`

[4] OpenSSL Software Foundation: *EVP - OpenSSLWiki [online].* červenec 2016, [cit. 2017-05-04]. Dostupné z: `https://wiki.openssl.org/index.php/EVP`

[5] Elliptic Curve Cryptography (ECC) Certificates Performance Analysis [online]. Technická zpráva, 2013, [cit. 2017-05-04]. Dostupné z: `https://www.symantec.com/content/en/us/enterprise/white_papers/b-wp_ecc.pdf`

[6] Singleton Pattern [online]. [cit. 2017-05-04]. Dostupné z: `http://www.oodesign.com/singleton-pattern.html`

[7] Dotson, J.: HTTP vs. HTTPS: What's the Difference? [online]. Červenec 2007, [cit. 2017-05-09]. Dostupné z: `http://www.biztechmagazine.com/article/2007/07/http-vs-https`

[8] R. Fielding, J. R.: Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing [online]. Technická zpráva, Červen 2014, [cit. 2017-05-04]. Dostupné z: `https://tools.ietf.org/html/rfc7230`

[9] Point, T.: HTTP - Responses. [online], 2017, [cit. 2017-05-09]. Dostupné z: https://www.tutorialspoint.com/http/http_responses.htm

[10] What is OSI Model & the Overall Explanation of ISO 7 Layers. [online], Březen 2017, [cit. 2017-05-05]. Dostupné z: http://nhprice.com/wp-content/uploads/2013/03/1-Tutorial-OSI-7-layer-model.jpg

[11] Grigorik, I.: Transport Layer Security (TLS). [online], 2013, [cit. 2017-05-05]. Dostupné z: https://hpbn.co/assets/diagrams/9873c7441be06e0b53a006aac442696c.svg

[12] Grigorik, I.: Transport Layer Security (TLS). [online], 2013, [cit. 2017-05-05]. Dostupné z: https://hpbn.co/assets/diagrams/b83b75dbbf5b7e4be31c8000f91fc1a8.svg

[13] OpenSSL Software Foundation: *SSL_get_verify_result [online]*. [cit. 2017-05-05]. Dostupné z: https://www.openssl.org/docs/man1.0.2/ssl/SSL_get_verify_result.html

[14] Villanueva, J. C.: What Is HMAC And How Does It Secure File Transfers? [online]. Srpen 2016, [cit. 2017-05-11]. Dostupné z: http://www.jscape.com/blog/what-is-hmac-and-how-does-it-secure-file-transfers

[15] Grigorik, I.: Transport Layer Security (TLS) [online]. 2013, [cit. 2017-05-11]. Dostupné z: https://hpbn.co/transport-layer-security-tls/

[16] OpenSSL Software Foundation: *EVP - OpenSSLWiki [online]*. [cit. 2017-05-06]. Dostupné z: https://wiki.openssl.org/index.php/Manual:Pem(3)

[17] OpenSSL Software Foundation: *BIO [online]*. [cit. 2017-05-06]. Dostupné z: https://www.openssl.org/docs/man1.0.2/crypto/bio.html

[18] OpenSSL Software Foundation: *SSL_CTX_new [online]*. [cit. 2017-05-09]. Dostupné z: https://www.openssl.org/docs/man1.0.2/ssl/SSL_CTX_new.html

[19] Cisco: Certificate Requirements for TLS [online]. , č. 1706, 2015, [cit. 2017-05-07]. Dostupné z: https://documentation.meraki.com/zGeneral_Administration/Other_Topics/Certificate_Requirements_for_TLS

[20] IBM: Key usage extensions and extended key usage [online]. Srpen 2008, [cit. 2017-05-07]. Dostupné z: https://www.ibm.com/support/knowledgecenter/en/SSKTMJ_8.0.1/com.ibm.help.domino.admin.doc/DOC/H_KEY_USAGE_EXTENSIONS_FOR_INTERNET_CERTIFICATES_1521_OVER.html

[21] Nguyen, J.: *OpenSSL Certificate Authority [online]*. Prosinec 2015, [cit. 2017-05-06]. Dostupné z: `https://jamielinux.com/docs/openssl-certificate-authority/`

Příloha A

Seznam použitých zkratek

API Application Programming Interface

ASCII American Standard Code for Information Interchange

Base64 Kódování binárního obsahu do textu

ČVUT České vysoké učení technické v Praze

DER Distinguished Encoding Rules

ECDSA Elliptic Curve Digital Signature Algorithm

FIT Fakulta informačních technologií

HMAC Keyed-hash Message Authentication Code

HTTP Hypertext Transfer Protocol

HTTPS HTTP Over TLS, Hypertext Transfer Protocol Secure

IP Internet Protocol

MAC Message Authentication Code

OS Operating system

PEM Formát uložení dat založen na Base64/ASCII

PKI Public Key Infrastructure

RFC Request for Comments

RSA Kryptografický algoritmus – asymetrická šifra

TLS Transport Layer Security

TPM Trusted Platform Module

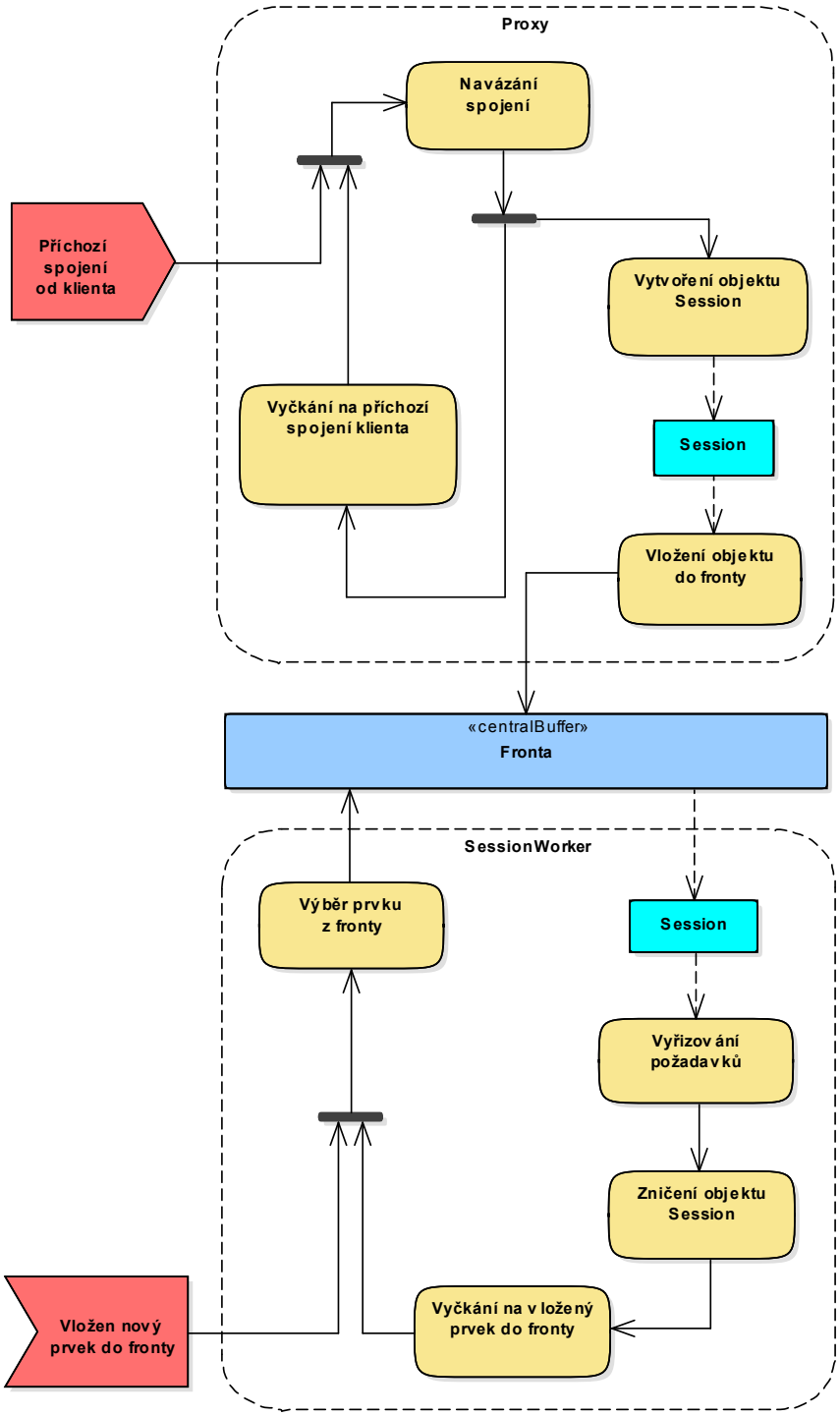
Příloha **B**

Obsah přiloženého CD

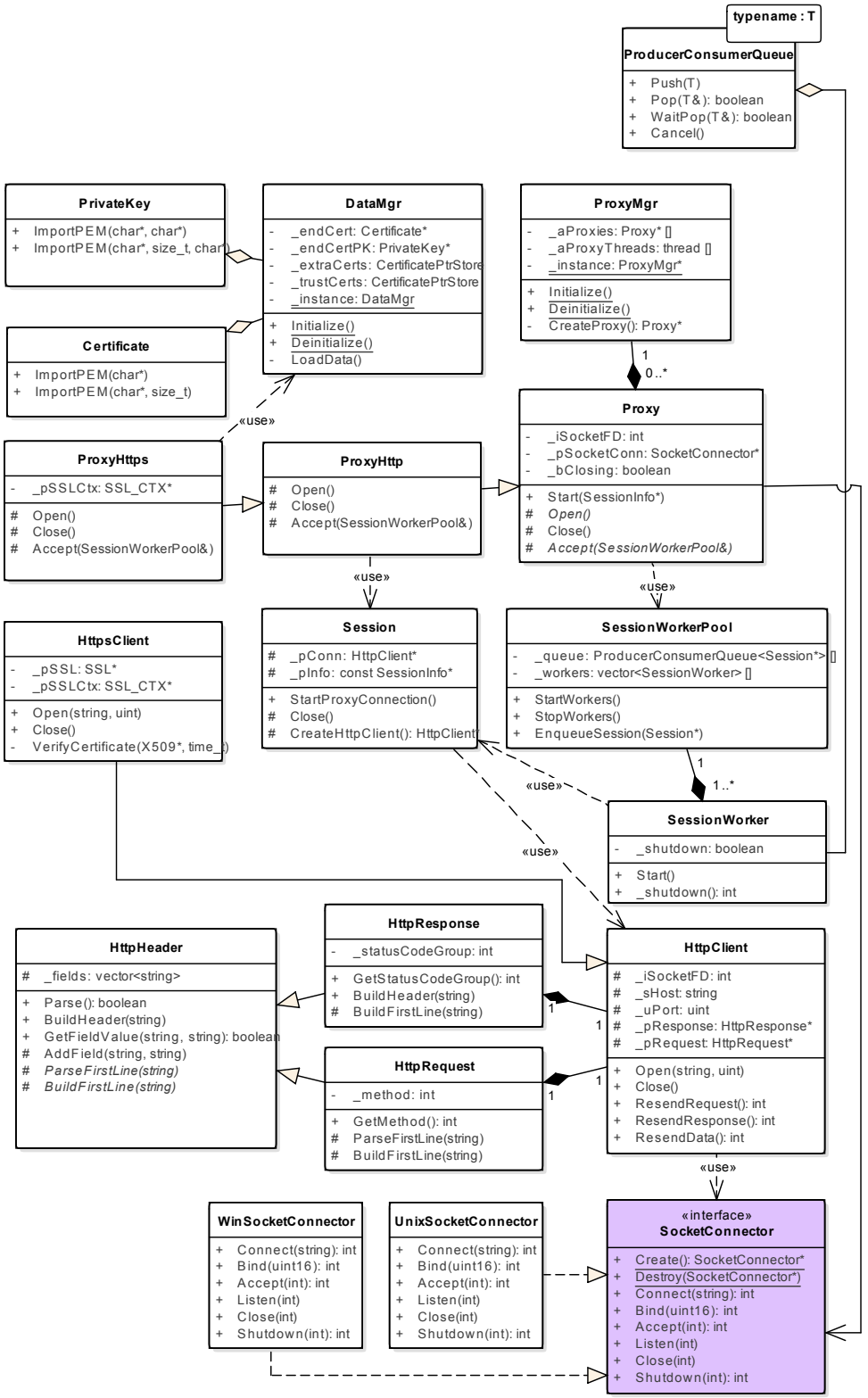
- `readme.txt` ..... stručný popis obsahu CD
- `app` ..... adresář obsahující snapshot vývojového repozitáře
- `etc`
  - `boot` ..... skript pro zavedení proxy do startu systému
  - `ca` ..... skripty pro vydání certifikátů
  - `certs` ..... vygenerované certifikáty pro doménu progtest.fit.cvut.cz
  - `keys` ..... privátní klíče k vygenerovaným certifikátům
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojové soubory práce ve formátu LaTeX
  - `uml-class-diagram.png` ..... UML diagram tříd implementace
- `text`
  - `BP_Šmíd_Tomáš_2017.pdf` ..... text práce ve formátu PDF

Příloha C

Přiložené obrázky



Obrázek C.1: Fronta producent-konzument



Obrázek C.2: UML diagram tříd (výřez)

Příloha D

Přiložené ukázky

```
void SessionWorker::Start()
{
   LOG_TRACE("SessionWorker::Start");

   if (!_queue)
      return;

   while (true)
   {
      Session* t = NULL;

      _queue->WaitPop(t);

      if (_shutdown || !t)
         break;

       t->StartProxyConnection();

      delete t;
   }
}
```

Ukázka D.1: Zdrojový kód funkce Start – vybírání objektů Session z fronty

```
-----BEGIN CERTIFICATE-----
MIIFgDCCA2igAwIBAgIJAMh/NbXyKRz/MA0GCSqGSIb3DQEBCwUAME0xCzAJBgNV
BAYTAkNaMRcwFQYDVQQIDA5DemVjaCBSZXB1YmxpYzERMA8GA1UECgwIRklUIENW
VVQxEjAQBgNVBAMMCVJvb3QgVGVzdDAeFw0xNzAyMDUwMzIzMzNaFw0zNzAxMzEw
MzIzMzNaME0xCzAJBgNVBAYTAkNaMRcwFQYDVQQIDA5DemVjaCBSZXB1YmxpYzER
MA8GA1UECgwIRklUIENWVVQxEjAQBgNVBAMMCVJvb3QgVGVzdDCCAiIwDQYJKoZI
hvcNAQEBBQADggIPADCCAgoCggIBALtmtL1XtJsZd0MjhDLVIrWgLV9lshjYhkh/
Ltyj6F8unS9SOcPSpvw76IujkaYGmoHrbw9rNwDEc4rh655LRPc3c2hlicnJtmeT
+5/rUkcxTURv5x4sncE/Q9h5Gvr8I85eWskfvNaZSZEoUvKwKbt/jZLjuTlntsa6
CGJs+EYbTL5gJxEX7MJO9eCNNZ2RvLzmOaKJMxL/noVbOcKZCirs149Tz+GnEFxj
eP1ME4muEXpyqcieKqH/hX15HsR4tzIB1SZEmkSK9HxbjBTDRrtlQxDZkb0MdOTx
NT22e7nFamXPOUwA1yv3YG3i+bolRFWGFkoNb4kghdYu2JedtYLGma24IN3DxNo0
BslA7i/dC0SnPDvnM7fNxLwYVnesmX91KTxaeIaZwhcwVijMNkTNHdHZcZskqMve
HUY6OE4xYYzGee5OENXJcZ+7qrD29skJ5kQkDGhIkHX5lVY6z1LFq8YgIjQow4OJ
XFjK7wJnhUpXIpxB1yHnJ2wD1BdVY6bXJmMJn680ImbSV241UEtd/9/c+CVSDM7h
X2lM1iVJ4Khdf+PT/y/foNL9GkIXp3cFAnfOjhM98zidNdvbTuXl+vCBGLvJupSn
0fETVVfluivR4ffkwPJEXnBRzLt0c779w3AMg2o7Rm5qI3hGPOpibQqmSMHEcaj5
f7enqb8JAgMBAAGjYzBhMB0GA1UdDgQWBBQgMoDwURlVBLIiPt+p73FlJs09bjAf
BgNVHSMEGDAWgBQgMoDwURlVBLIiPt+p73FlJs09bjAPBgNVHRMBAf8EBTADAQH/
MA4GA1UdDwEB/wQEAwIBhjANBgkqhkiG9w0BAQsFAAOCAgEAj1Ny6hVcOo4/ovf/
ul2DC9qEV0dWUccv7FN6c4tZxufHR3OJZWDYJK5oZ4fuLBIyLMWdrQBgfLix1T3R
lsXV6OXATDsDGOGR6mnCAzPAC1WFs34PwuH5TLEJuuruJCfgMTaRV+xeSMG/RN88
lVq2nyGKRatUxdtlRD9EdglyVweZ6exuzV/XVn/ChhFXZLdYTq7YD4XvDYexwK15
m86Yo8l3McaPUrkmS4FiSQmC/M9bKUAvRfCQLY2mWhOt1h3DTP4O5g9OuG2yuOdf
c9x78mrabri4LXDyJ47rAIcVa2CHeTgX/wZJ5sH7V/GkIXhNAXlYthzLU5Qjx87P
6iPSBZl34sOT6cyJ3pH0f/73m/sXPGGC7i2UzRaI4Y7gW0p/lA+U/opzlqhlQRXE
KpAF+9/GuekTpRPvAhBA35p3tTtHymPWOWd87v9yZZ+w8KQiDJjzDOUEXfDqCa37
UAAX5YMYixryt3KySPJcJrpHwNE6v6QoLtoqnfYy97Ah5UsoyncbKEI5KumpNil9
0gJdz51PQO1/jcSZyVZglgRJXtHtr3V8D+f3ICMwhbmhw08GJ+RFj2NkYlJT0+aE
MeL2SL23PeO1KEyEMax/rkojAajO9uYn7lKPmmJWnkjA9yFmD1wyf3CJN81LekkA
sbb8cegMsquqPE8Qb7mjwhIcL8o=
-----END CERTIFICATE-----
```

Ukázka D.2: Kořenový certifikát v PEM formátu