VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

# DIPLOMOVÁ PRÁCE

Brno, 2021 Bc. Václav Šnajdr


# 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


# WEBOVÁ APLIKACE PRO TESTOVÁNÍ ZRANITELNOSTÍ WEBOVÉHO SERVERU
WEB APPLICATION FOR TESTING WEB SERVER VULNERABILITIES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS


AUTOR PRÁCE
AUTHOR
Bc. Václav Šnajdr

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. David Smékal


BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

# Diplomová práce

magisterský navazující studijní program **Informační bezpečnost**

Ústav telekomunikací

***Student:*** Bc. Václav Šnajdr ***ID:*** 195173
***Ročník:*** 2 ***Akademický rok:*** 2020/21

**NÁZEV TÉMATU:**

## Webová aplikace pro testování zranitelností webového serveru

**POKYNY PRO VYPRACOVÁNÍ:**

Cílem diplomové práce je návrh a implementace webové aplikace realizující online kontrolu zranitelnosti vzdáleného webového serveru.
V rámci práce bude navrženo uživatelské rozhraní aplikace (GUI), webová prezentace a implementace automatických Python skriptů pro online vyhodnocení úrovně bezpečnosti serveru.
Navržená aplikace (s využitím Nette, Bootstrap) bude plně funkční a bude demonstrovat princip a vlastnosti navrženého řešení. Aplikace bude testovat minimálně 5 různých zranitelností. V diplomové práci bude realizováno experimentální prostředí pro prezentaci funkčnosti automatických skriptů a vyhodnocení úrovně bezpečnosti včetně grafického znázornění.

**DOPORUČENÁ LITERATURA:**

[1] CASTRO, Elizabeth a Bruce HYSLOP. HTML5 a CSS3: názorný průvodce tvorbou WWW stránek. Brno: Computer Press, 2012, 439 s.ISBN 978-80-251-3733-8.

[2] RISTIC, Ivan. Bulletproof SSL and TLS: Understanding and Deploying SSL/TLS and PKI to Secure Servers and Web Applications. Feisty Duck, 2013, ISBN: 978-1907117046.

***Termín zadání:*** 1.2.2021 ***Termín odevzdání:*** 24.5.2021

***Vedoucí práce:*** Ing. David Smékal

**doc. Ing. Jan Hajný, Ph.D.**
předseda rady studijního programu

## ABSTRAKT

Diplomová práce se zabývá návrhem a implementaci webové aplikace pro testování bezpečnosti SSL/TLS protokolů na vzdáleném serveru. Webová aplikace je vyvíjena ve frameworku Nette. V teoretické části jsou popsány SSL/TLS protokoly, zranitelnosti, doporučení a použité technologie v praktické části. Praktická část se věnuje tvorbě webové aplikace s procesem použití automatických skriptů pro testování a zobrazení výsledků na webové stránce s hodnocením A+ až C. Webová aplikace zároveň zobrazí seznam detekovaných zranitelností a jejich doporučení.



## KLÍČOVÁ SLOVA

HTTP, HTTPS, SSL, TLS, Nette, Bootstrap, Webová aplikace, testování, zranitelnosti, CRIME, Renegotiation, BEAST, Lucky13, Heartbleed, POODLE, Logjam, DROWN, Raccoon, HSTS, MVC, OWASP, OpenSSL, Nmap, NIST, ENISA, NÚKIB



## ABSTRACT

The Master's Thesis deals with the design and implementation of a web application for testing the security of SSL/TLS protocols on a remote server. The web application is developed in the Nette framework. The theoretical part describes SSL/TLS protocols, vulnerabilities, recommendations and technologies used in the practical part. The practical part is devoted to the creation of a web application with the process of using automatic scripts to test and display the results on the website with a rating of A+ to C. The web application also displays a list of detected vulnerabilities and their recommendations.



## KEYWORDS

HTTP, HTTPS, SSL, TLS, Nette, Bootstrap, Web application, testing, vulnerabilities, CRIME, Renegotiation, BEAST, Lucky13, Heartbleed, POODLE, Logjam, DROWN, Raccoon, HSTS, MVC, OWASP, OpenSSL, Nmap, NIST, ENISA, NÚKIB

ŠNAJDR, Václav. *Webová aplikace pro testování zranitelností webového serveru*. Brno, 2021, 109 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. David Smékal

# PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Webová aplikace pro testování zranitelností webového serveru“ jsem vypracoval 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 uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom 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

# PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Davidovi Smékalovi za odborné vedení, konzultace a podnětné návrhy k práci. Dále bych chtěl poděkovat své rodině za podporu během studia a partnerce Janě Ráboňové za trpělivost, porozumění a podporu při tvorbě této diplomové práce.

# Obsah

# Seznam obrázků

# Seznam tabulek

# Seznam výpisů

# Úvod

V současné době je bezpečnost přenášených dat velice důležitá. Ve světě internetu, do kterého se často lidé připojují a komunikují pomocí prohlížeče, se přenáší i citlivá data, přihlašovací údaje nebo platební informace. Citlivá data by potenciální útočník mohl zneužít a v nejhorším případě ukrást identitu uživatele.

Pro zajištění bezpečnosti v rámci internetu jsou využívány SSL/TLS protokoly, které pracují na transportní vrstvě a dokážou zajistit důvěrnost, integritu a autentizaci přenášených dat. Avšak i přesto existují známé útoky, které zneužívají známé zranitelnosti u jednotlivých verzí SSL/TLS protokolů. Proto je nutné používat doporučené a podporované verze.

Diplomová práce se věnuje návrhu webové aplikace pro testování bezpečnosti vzdáleného webového serveru. Implementace webové aplikace obsahuje automatizované python skripty, které zjistí informace o poskytovateli vzdáleného serveru, stavy známých portů na běžícím serveru, verze podporovaných TLS protokolů s kryptografickými balíčky a informace o použitém certifikátu. Webová aplikace na základě zjištěných informací poskytne vyhodnocení testovaného vzdáleného serveru se známkou od A+ až C, dále poskytne seznam detekovaných zranitelností a nabídne ke konkrétní zranitelnosti doporučení. Webová aplikace je vyvíjena ve frameworku Nette a stylizována ve frameworku Bootstrap.

# 1 Teoretická část

V teoretické části diplomové práce je vysvětlen způsob zabezpečení komunikace v síti internetu. Konkrétně jde o přístupy na webové servery. Dále jsou vysvětleny technologie použité v praktické části, zranitelnosti obsažené v různých verzích SSL/TLS protokolů a nástroje, kterými lze webový server otestovat.

## 1.1 Algoritmy používané pro zabezpečení spojení

V této podkapitole jsou popsány algoritmy, které jsou použité pro zabezpečená spojení. Hlavní způsoby použití algoritmů pro šifrování, které jsou obsahem této práce, se dělí na:

- **Asymetrické šifry** - obsahují veřejný klíč a soukromý klíč. Veřejný klíč je k dispozici pro šifrování ze strany odesílatele zprávy. Soukromý klíč slouží pro dešifrování a má ho pouze příjemce, nikdo jiný. Veřejný klíč je vypočítán ze soukromého klíče. Odvození soukromého klíče pomocí veřejného je nemožné, pokud se vybrala vhodná délka klíče.
- **Symetrické šifry** - obsahují pouze jeden klíč, kterým lze šifrovat a dešifrovat zprávu. Zde se předpokládá, že strany mají tento klíč již k dispozici. Jsou rychlé oproti asymetrickým šifrám při šifrování velkého objemu dat.
- **Algoritmy pro ustanovení klíče** - tyto algoritmy dokážou sestrojit symetrický klíč pro obě strany [1].

### 1.1.1 Šifrovací algoritmy

Použité algoritmy asymetrického šifrování jsou následující:

- **DSA (Digital Signature Algorithm) algoritmus** - navržen institucí NIST (Národní institut standardů a technologie) v roce 1991 a byl uvedený ve standardu FIPS 186-4. DSA slouží pro generování digitálního podpisu a zaručuje nepopíratelnost odesílatele zprávy. Existuje i DSA nad eliptickými křivkami označen jako ECDSA (Elliptic Curve Digital Signature Algorithm) [2].
- **RSA (Rivest, Shamir, Adleman) algoritmus** - používá se pro šifrování nebo také pro podepisování zpráv. Jeho bezpečnost je založena na obtížnosti rozkladu neboli faktorizaci velkého čísla $n$ na dvě prvočísla $p$ a $q$ tak, aby platilo $n = p \cdot q$. Název tohoto algoritmu je složen ze tří autorů R.L. Rivest, A. Shamir a L. Adleman [3].
- **Diffie-Hellman algoritmus (DHE)** - algoritmus nebo také protokol DHE spadá do kategorie asymetrických šifer. Využívá se pro ustanovení klíče během nezabezpečeného spojení. Strany si navzájem přenesou veřejné parametry,

z kterých si každá strana spočítá tajný klíč. Tento klíč je stejný na obou stranách. Tím je spojení připraveno pro symetrické šifrování. Bezpečnost Diffie-Hellman algoritmu je založena na problému diskrétního logaritmu [4][5].

- **ECDHE (Elliptic-curve Diffie–Hellman)** - jedná se o algoritmus Diffie-Helman s využitím eliptických křivek. V současné době se jedná o nejvyužívanější protokol pro zabezpečené spojení [6].

Nejběžnější symetrické algoritmy pro šifrování, které lze nalézt v této práci, jsou uvedeny zde:

- **RC4 (Rivest Cipher 4)** - proudová symetrická šifra, vyvinutá v roce 1987 Ronem Rivestem, vytváří dlouhé sekvence klíčů a přidává je do datových bajtů. Data šifruje pomocí operace XOR bajt po bajtu s klíčem [7].
- **3DES (Data Encryption Standard)** - bloková symetrická šifra DES byla standardizována v roce 1977. Obsahuje šestnáct iterací a pro každou iteraci se vypočítá 48bitový klíč (round key), kterým se šifruje 32bitový blok zprávy [8]. Původní standard DES obsahuje délku klíče 56 bitů, která je velmi ohrožena proti útoku hrubou silou. Proto byl navržen 3DES a znamená trojnásobnou aplikaci symetrické šifry DES. První aplikace zašifruje data prvním klíčem, druhá aplikace data dešifruje druhým klíčem a třetí zašifruje posledním klíčem. Tento postup je označován jako EDE (Encrypt-Decrypt-Encrypt) [9][10].
- **AES (Advanced Encryption Standard)** - nástupcem DES je standard AES. Bloková symetrická šifra, která podporuje tři délky klíče 128, 192 a 256 bitů. Délka se uvádí v názvu, např. AES-256 [8].

Blokové šifry používají pro své operace módy. Ty řeší problém rozdílných velikostí zpráv do pevně dané délky bloku [8]. Nejběžněji se lze setkat s těmito módy:

- **CBC (Cipher Block Chaining)** - při použití CBC módu jsou bloky šifrovány s předchozími bloky šifrovaného textu. CBC vyžaduje náhodný a nepředvídatelný inicializační vektor (IV), který se kombinuje s prvním blokem prostého textu [11].
- **CTR (Counter)** - mód CTR odstraňuje závislost bloku na jiný blok a využívá čítač, který se mění po každé iteraci. Všechny čítače musí být navzájem odlišné. Šifrování módem CTR lze paralelizovat a chybný blok je možné obnovit bez ovlivnění ostatních bloků [11].
- **CCM (Counter with Cipher Block Chaining-Message Authentication Code)** - tento mód kombinuje čítač s autentizačními prvky, a tak zajišťuje šifrování a autentizaci během přenosu dat [12].
- **GCM (Galois/Counter Mode)** - GCM je režim pro algoritmus AES. Obsahuje čítač pro zajištění důvěrnosti dat a ověření těchto dat pomocí univerzální hašovací funkce, která je definována přes binární konečná pole neboli Galois

pole. GCM lze paralelizovat a dokáže detekovat náhodné, úmyslné nebo neoprávněné úpravy dat [13].

### 1.1.2 Hašovací funkce

Hašovací funkce je algoritmus, který převádí vstupní data libovolné délky na výstup s pevnou délkou. Výsledek je označován jako haš a má tyto vlastnosti:

- Z vypočítaného haše je výpočetně nemožné najít nebo zkonstruovat původní zprávu nebo vzor.
- Ze známé zprávy a vypočítaného haše je výpočetně nemožné najít jinou zprávu se stejným hašem.
- Je výpočetně nemožné najít dvě libovolné zprávy, které mají stejný haš.

Jinak řečeno, hašovací funkce vyžadují jednosměrnost a bezkoliznost.

Nejčastěji jsou hašovací funkce používány pro rychlé porovnávání velkého množství dat, ověřování zpráv, digitální podpisy nebo slouží jako otisk prstu (fingerprint) [14][15]. Nejznámějšími hašovacími algoritmy jsou:

- **MD5 (Message-Digest algorithm)** - vytváří 128bitový haš zprávy a používá se spíše jako kontrolní otisk. V současné době se považuje za nebezpečný, např. při použití k uchování hesel, kdy se heslo převádí na hašový tvar a v tomto tvaru je uložen [16][17].
- **SHA-1 (Secure Hash Algorithm)** - tato funkce vytvoří haš o velikosti 160 bitů a od roku 2017 je považován za prolomený [14][18].
- **SHA-2** - národní institut standardů a technologie (NIST) definovala bezpečnější verze SHA algoritmů s velikostí 224, 256, 384 a 512 bitů. Často se velikosti uvádějí v názvu, např. SHA-256. V současné době jde o bezpečné algoritmy [19].
- **SHA-3** - v současné době nejnovější rodina hašovacích algoritmů standardizována v roce 2015 institucí NIST na základě vyhrané soutěže o nového nástupce funkcí SHA-1 a SHA-2. SHA-3 je také známý pod svým původním jménem **Keccak**. Rodina SHA-3 je složena z funkcí SHA3-224, SHA3-256, SHA3-384 a SHA3-512. Také obsahuje funkce SHAKE128 a SHAKE256, které se liší od hašovacích funkcí, ale lze je využít podobným způsobem jako hašovací funkce s flexibilitou jednotlivých požadavků aplikací [20].

## 1.2 Protokoly pro zabezpečené spojení

V 90. letech si firma Netscape uvědomila nutnost zabezpečit síťový tok na internetu, a tak vytvořila protokol SSL (Secure Sockets Layer). Cílem protokolu bylo vytvořit

šifrovaný provoz mezi klientem a serverem s nezávislým operačním systémem. Během vývoje vznikl protokol TLS (Transport Layer Security), který nahradil starý SSL s nedostačujícím zabezpečením. V současné době s nejnovější verzí je protokol TLS 1.3 [21].

Protokoly SSL/TLS jsou kryptografické protokoly, které zajišťují zabezpečenou komunikaci přes nezabezpečenou infrastrukturu na transportní vrstvě modelu TCP/IP. Přesněji lze vystihnout použití v modelu OSI (Open Systems Interconnection) v 6. prezentační vrstvě. Ve skutečnosti mají SSL/TLS protokoly zavedeny tyto čtyři cíle:

- **Kryptografické zabezpečení** - hlavní cíl protokolu. Umožňuje bezpečnou komunikaci mezi jakýmikoli dvěma stranami při výměně informací.
- **Interoperabilita** - nezávislí vývojáři by měli být schopni vyvíjet produkty, které jsou schopné vzájemně komunikovat pomocí běžných kryptografických parametrů.
- **Rozšiřitelnost** - být nezávislý na skutečně použitých kryptografických primitivech jako jsou šifry nebo hašovací funkce. Tím je dosažena migrace z jednoho primitivu na druhý, aniž by se musel vytvářet nový protokol.
- **Účinnost** - posledním cílem je dosáhnout všech předchozích cílů při přijatelných nákladech na výkon. Snížit náklady kryptografických operací na minimum, poskytnout schématu ukládání do mezipaměti relace a vyhnout se následujících připojení [15].

### 1.2.1 Verze SSL protokolu

Starý protokol SSL vyšel ve třech verzích. Nyní jsou tyto verze označeny za velice zranitelné a obsahují chyby.

#### SSL 1.0

Tato verze nebyla vůbec oficiálně publikována [21].

#### SSL 2.0

Druhá verze byla vydána v roce 1995 a zakázána až v roce 2011. Obsahoval slabý MD5 haš a byl zranitelný proti útoku MITM (Man-in-the-middle), v překladu „muž uprostřed“, který vynutil použití slabých kryptografických balíčků [22].

#### SSL 3.0

Poslední verze protokolu vydána v roce 1996. Následně byla ukončena podpora v roce 2015 z důvodu nedostatečné bezpečnosti v módu CBC (Cipher block chaining) nebo

šifry RC4. Dále je zranitelný proti útoku POODLE (Padding Oracle On Downgraded Legacy Encryption) [23][24].

### 1.2.2 Verze TLS protokolu

Protokol TLS je vydán ve čtyřech verzích. Opravuje závažné zranitelnosti a slabé kryptografické algoritmy.

#### TLS 1.0

První verze TLS 1.0 byla vydána v roce 1999. Oproti předchozímu protokolu SSL 3.0 neobsahoval velké změny [15]. Jeho podpora byla ukončena v roce 2018. Jeden z důvodů, proč již není podporován, je použití prolomeného hašovacího algoritmu SHA-1 [18][25].

#### TLS 1.1

Další verze TLS 1.1 byla vydána v roce 2006 a obsahovala některá rozšíření oproti předchozí TLS 1.0. Podpora pro tento protokol také skončila v roce 2018 společně s předchozí verzí ze stejných důvodů [15][25].

#### TLS 1.2

Běžně používaná v dnešní době je verze TLS 1.2 vydána v roce 2008. Byla přidána podpora pro ověřené šifrování a protokol se stal plně flexibilním díky odstranění pevně zakódovaných bezpečnostních primitiv [15][26].

Na obrázku 1.1 je znázorněn průběh navazování spojení a výměna zpráv protokolu TLS 1.2. Klient odešle uvítací zprávu **ClientHello**, na kterou musí server odpovědět uvítací zprávou **ServerHello**, jinak se připojení nenaváže. Uvítací zprávy budou dále v práci uváděné v anglickém jazyce. Obsahem těchto dvou zpráv jsou následující atributy:

- **Verze protokolu**,
- **ID relace**,
- **Šifrovací sada**,
- **Metoda komprese**.

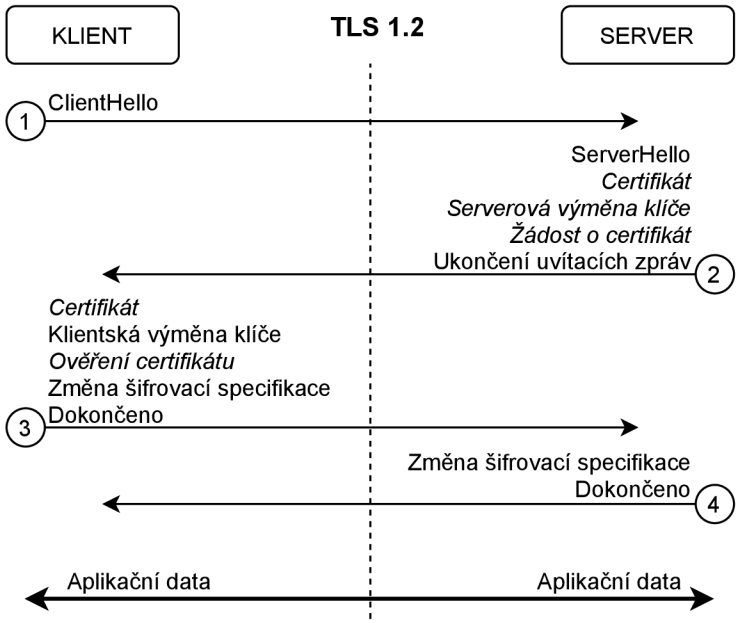
Také jsou generovány dvě náhodné hodnoty, které slouží pro výpočet klíče:

- **ClientHello.random**,
- **ServerHello.random**.

Pro výměnu klíčů se používají zprávy **Certifikát** u serveru a klienta, **Serverová výměna klíče** a **Klientská výměna klíče**. Díky těmto zprávám jsou schopny

se obě strany dohodnout na sdíleném tajemství. Toto tajemství musí být dostatečně velké s minimální délkou 46 bajtů [26].

Server pošle svůj certifikát, pokud má být ověřen, nebo místo toho může být odeslána zpráva **Serverová výměna klíče**. Tato zpráva je požadována buď z důvodu, že žádný certifikát server nemá, nebo je jeho certifikát určen pro podepisování. Pokud je server ověřen, může na základě vybrané šifrovací sady požadovat certifikát od klienta. Dále server pošle zprávu **Ukončení uvítacích zpráv**, která ukončuje výměnu uvítacích zpráv a čeká na odpověď klienta. Jestliže server poslal zprávu **Žádost o certifikát**, klient musí poslat zprávu se svým certifikátem. Klient poté odešle zprávu **Klientská výměna klíče**, která závisí na vybraném algoritmu veřejného klíče při komunikaci mezi **ClientHello** a **ServerHello** zprávami. Jestliže klient poslal certifikát s možností podepisování, odešle se také digitálně podepsaná zpráva **Ověření certifikátu**, která explicitně ověří držení soukromého klíče v certifikátu [26].




Obr. 1.1: TLS 1.2 Handshake proces.

Dále je klientem odeslána zpráva **Změna šifrovací specifikace** a zpráva **Dokončeno**. V odpovědi server odešle vlastní zprávu **Změna šifrovací specifikace** a také zprávu **Dokončeno**. Nyní je dokončen celý proces potřesení rukou neboli anglicky handshake a strany si mohou vyměňovat data aplikační vrstvy [26]. V práci bude termín handshake uváděn v anglickém jazyce.

Zprávy **Změna šifrovací specifikace** obsahují informace ohledně generování prostředků pro klíč, algoritmus pro šifrování dat a MAC (Message authentication code) algoritmus pro autentizaci [26].

Některé zprávy uvedené na obrázku 1.1 jsou volitelné nebo jsou závislé na dané situaci a nemusejí být vždy odeslány. Tyto zprávy jsou napsány kurzívou [26].

### TLS 1.3

Nejnovější verze je TLS 1.3, která byla vydána v roce 2018, zrychluje navazování spojení a je zbavena slabých blokových šifer a hašovacích funkcí. Podporuje tři základní režimy výměny klíčů:

- **(EC)DHE** - Diffi-Hellman přes konečná pole nebo eliptickými křivkami,
- **PSK-only** - pouze s předsdíleným klíčem,
- **PSK with (EC)DHE** - předsdílený klíč s Diffi-Hellman algoritmem.

Navazování spojení neboli handshake má tři fáze:

- **Key Exchange (Výměna klíče)** - vytvoří se sdílené klíčovací materiály a vyberou se kryptografické parametry. Po této výměně je vše šifrováno.
- **Server parameters (Serverové parametry)** - stanoví se další parametry handshake spojení např. podpora protokolů aplikační vrstvy, ať už je, nebo není klient ověřen.
- **Authentication (Ověření)** - ověření serveru, případně i klienta. Strany si poskytnou navzájem klíčové potvrzení a integritu handshake spojení [27].

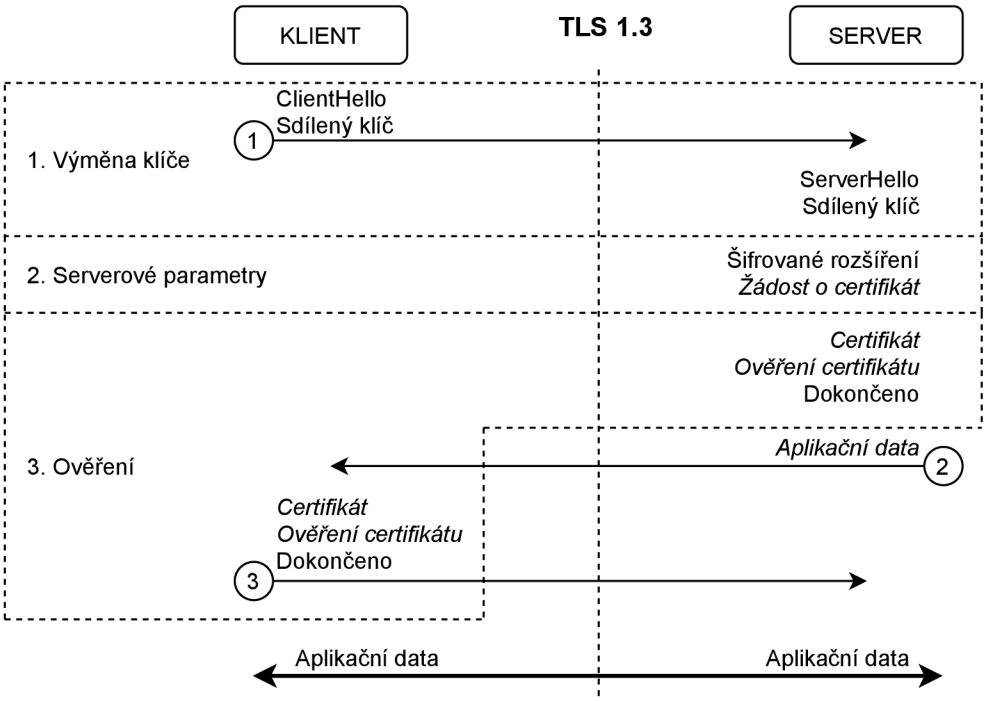
Protokol TLS 1.3 také dokáže ze strany serveru po klientské uvítací zprávě **ClientHello** odesílat aplikační data ještě před tím, než je klient autentizován. Průběh handshake spojení je zobrazen na obrázku 1.2. Taktéž zprávy napsané kurzívou nemusejí být vždy odeslány [27].

## 1.3 Zabezpečení webové stránky

Protokol HTTP (Hypertext Transfer Protocol) se používá na aplikační vrstvě modelu TCP/IP, který umožňuje aplikacím komunikovat s dalšími vrstvami potřebné k přenosu dat přes internet. Přenáší čitelná data v komunikaci *žádost-odpověď* webového prohlížeče s webovým serverem. Neobsahuje žádné zabezpečení, které by zabránilo přečíst data procházející internetem [28].

V současné době se protokol HTTP používá v mnoha aplikacích, a tak je potřeba chránit přenášený obsah. Pomocí kombinací HTTP a TLS tím bylo dosaženo. Vznikl tak protokol HTTPS (Hypertext Transfer Protocol Secure), který chrání přenos webového obsahu, a tudíž je zajištěna jeho důvěrnost, integrita a autentizace. Zatímco HTTP běží na portu 80, zabezpečený HTTPS běží na portu 443 [29].

Útočníkovi je tedy zabráněno číst přenášená data, někdy i citlivá data, jako jsou hesla. Avšak existují některé zranitelnosti, které útočník dokáže zneužít a tím se




Obr. 1.2: TLS 1.3 Handshake protocol.

dostat do kritických míst. Majetek, informace nebo cokoliv, co je cenné, se nazývají aktiva a k těm se útočník často chce dostat.

## 1.4 Zranitelnosti a útoky

Zranitelnosti v HTTPS často spočívají v chybné implementaci starého protokolu, nedodržení doporučení, nebo používání slabých šifrovacích algoritmů. Díky tomu lze zrealizovat útoky, které těchto zranitelností využijí.

### 1.4.1 CRIME Attack

Zranitelnost, kterou využívá CRIME (Compression Ratio Info-leak Made Easy) útok, spočívá v používání kompresí v TLS spojení. V roce 2012 přišlo praktické využití postranních kanálů při kompresi. Pokud je komprese zapnutá, útočník v pozici MITM pomocí postranním kanálem dokáže získat HTTP záhlaví a sledovat tím rozdíly délek v hlavičce. Sledováním komprimovaných velikostí požadavků se nakonec odvodí obsah zbývající části požadavku, kterými mohou být soubory relace, a dosáhne se únosu spojení [15][30].

#### Obrana proti CRIME útoku

Účinnou obranou proti útoku CRIME je vypnout používání komprese na serveru na TLS úrovni [31].

### 1.4.2 Renegotiation

Opětovné vyjednávání neboli Renegotiation probíhá pokud předchozí TLS spojení ukončí komunikaci, ale strany požadují další komunikaci. Provede se nové potřesení rukou, kde se dohodnou nové parametry pro zabezpečené připojení. Chyba v zabezpečení týkající se opětovného vyjednávání byla objevena v roce 2009. Zjištění vychází z toho, že mezi starými a novými TLS spojení neexistuje kontinuita. Server tedy neověřuje, že za obě konverzace stojí stejná strana. Tuto chybu lze útočník typu MITM zneužít v následujících krocích:

1. Zachytit žádost o připojení TCP od oběti k cílovému serveru.
2. Otevřít nové spojení TLS k serveru a odeslat infikovaná data.
3. Pokračovat v přenosu jako transparentní proxy. Data od útočníka a oběti budou součástí stejného datového toku [15].

#### Obrana proti opětovnému vyjednávání

K prevenci zneužití opětovného vyjednávání je doporučeno zakázat opětovné vyjednávání iniciované klientem. Dále je doporučeno podporovat nový standard bezpečného opětovného vyjednávání **Secure Renegotiation**. Jedná se o rozšíření v TLS spojení, které dokládá znalost předchozího potřesení rukou mezi dvěma stranami [15].

### 1.4.3 BEAST Attack

Útok BEAST (Browser Exploit Against SSL/TLS) je technika, kterou lze použít proti protokolu TLS 1.0 a starším SSL. Byl oznámen v roce 2011 a spočívá na dřívější známé slabosti konstrukce IV. Tato slabina byla opravena ve verzi TLS 1.1, ale v té době neexistovala podpora v prohlížeči pro novější TLS. Útok se zaměřuje na šifrování v módu CBC implementované v protokolu a to na předvídatelnost IV [15].

#### Obrana proti BEAST útoku

Je doporučeno protokoly TLS 1.0 a starší SSL zakázat na webových serverech a tím zamezit možným útokům [25].

### 1.4.4 Lucky13

Útok Lucky13 byl publikován v roce 2013 a jedná se o útok s načasováním. Využívá chybu v použití CBC módu ve verzích TLS 1.2, TLS 1.1 a také ve starších verzích. Útok umožňuje úplné obnovení prostého textu v TLS relaci. Spíše jde o teoretický útok, který byl však prokázán v praxi [32][33].

#### Obrana proti útoku Lucky13

Obranou proti útoku Lucky13 je používat novější módy šifrování například mód GCM [34].

### 1.4.5 Heartbleed

V roce 2014 byla veřejnosti představena zranitelnost Heartbleed v nástroji OpenSSL, který obsahuje knihovny pro implementaci SSL/TLS protokolů. Jedná se o chybnou implementaci protokolu Heartbeat, která rozšiřuje protokol TLS. Heartbeat přidává kontrolní funkci, zda je druhá strana v konverzaci stále k dispozici. Zaměřuje se na DTLS (Datagram Transport Layer Security) protokol používající nespolehlivé UDP (User Datagram Protocol) spojení.

Útočník dokáže vzdáleně načíst až 64 kB procesní paměti serveru v jednom požadavku. Těchto požadavků může poslat více, a tak načíst neomezený počet dat včetně citlivých dat [15].

Této zranitelností lze předejít použitím správné verze OpenSSL [35].

### 1.4.6 POODLE Attack

Útok POODLE oznámil bezpečnostní tým v Googlu v roce 2014. Je použit v chybném zabezpečení protokolu SSL 3.0, která umožňuje útočníkům v síti číst části šifrovaného textu s nechráněnou výplní (padding) při použití módu CBC. Díky tomu mohou útočníci provádět změny výplně k odhalení šifrovaného obsahu. CBC mód ověřuje pouze prostý text, ale výplň ponechává nechráněnou.

Toto bylo opraveno v novějších verzích TLS protokolů [15].

### 1.4.7 Logjam Attack

Kryptografický algoritmus Diffe-Hellman (DH) je nepopulárnější při výměně klíčů. Je základem mnoha protokolů včetně HTTPS, které se spoléhají na TLS spojení. Útok Logjam umožňuje útočníkovi typu MITM degradovat zranitelná spojení TLS na 512b kryptografii pro export. Útočník poté může přenášena data číst a upravovat. Zranitelnost je způsobena chybou v TLS protokolu při výměně klíčů u DH

algoritmu. Všechny servery, které podporují šifrovací sady **DHE_EXPORT**, jsou proti tomuto útoku zranitelné [36].

#### Obrana proti Logjam útoku

Účinná obrana proti útoku Logjam je na webovém serveru zakázat šifrovací sady, které obsahují možnost exportování **DHE_EXPORT** a použít 2048b délku pro DH algoritmus [36][37].

### 1.4.8 DROWN Attack

DROWN (Decrypting RSA with Obsolete and Weakened encryption) útok byl zveřejněn v roce 2016 a uvádí se, že v té době bylo zranitelných 33 % webových HTTPS serverů. Zranitelnost serverů spočívá v podpoře staršího a špatně zabezpečeného protokolu SSL 2.0. Útočník může dešifrovat moderní TLS spojení komunikujících stran odesláním sondy na server, který podporuje SSL 2.0 protokol a používá stejný soukromý klíč. Mnoho serverů používá stejný certifikát a klíč, a pokud tedy jeden ze serverů používá nebezpečný protokol SSL 2.0, útočník díky tomu může napadnout i server, který tento protokol nepodporuje. Útočník při úspěšném útoku dokáže dešifrovat spojení a zjistit citlivá data, hesla nebo čísla kreditních karet [38].

#### Obrana proti DROWN útoku

Ochranu proti útoku DROWN lze zajistit tak, aby soukromé klíče na serverech nebyly použity se serverový softwarem, který umožňuje připojení přes starý protokol SSL 2.0. Software zahrnuje nejen webové servery, ale i poštovní servery, který podporují SSL/TLS zabezpečení. Další obranou je zakázání starého protokolu SSL 2.0, avšak pouhé zakázání protokolu může být komplikované a závisí na konkrétním softwaru serveru [38].

### 1.4.9 Raccoon Attack

Raccoon útok byl publikován v roce 2020 a je založen na chybě načasování v TLS protokolu. Za určitých podmínek dokáže útočník přerušit šifrování a číst citlivou komunikaci. Tuto chybu je těžké zneužít a spoléhá se na velmi přesné načasování a konkrétní konfigurace serveru.

Aby útok mohl být realizován, útočník musí být typu MITM v blízkosti serveru, cílový server musí podporovat šifrovací sadu s výměnou klíčů pomocí DH nebo DHE algoritmu a sever musí tyto sdílené klíče používat pro více připojení. Útočník tak dokáže určit sdílený klíč relace a použít jej k dešifrování komunikace. K odposlechu další komunikace musí tento útok provést znova [39][40].

#### Obrana proti Raccoon útoku

Přesto, že se jedná o náročně proveditelný útok, doporučuje se zakázat kryptografické sady **TLS_DH** a **TLS_DHE** a místo nich používat DH algoritmus nad eliptickými křivkami. V případě použití nejnovějšího protokolu TLS 1.3 je jednou z obranných možností, kde nehrozí zneužití DH nebo DHE algoritmů [39][40].

### 1.4.10 SSL Stripping Attack

Útok SSL Strip je technika, která degraduje zabezpečenou webovou stránku HTTPS na nezabezpečenou HTTP. Využívá se při útoku MITM, kdy útočník převede veškerý provoz přes sebe. Spojení mezi obětí a útočníkem je nezabezpečený, ale spojení mezi útočníkem a serverem je zabezpečený [41].

#### Obrana proti degradaci zabezpečeného spojení

Účinná obrana proti útoku SSL Strip je mechanismus HSTS (HTTP Strict Transport Security). Tento mechanismus vynutí použití zabezpečeného spojení přepsáním URL (Uniform Resource Locator), aby použilo šifrování a při neplatném nebo neověřeném certifikátu nepustí uživatele na daný web [15][42].

## 1.5 Doporučení a standardy

Existuje několik organizací a institucí, které vydávají doporučení nebo standardy k ochraně utajovaných informací v oblasti informačních a komunikačních systémů a kryptografické ochrany. Mezi ty nejznámější patří organizace NIST. Dalším je ENISA (Evropská agentura pro bezpečnost sítí a informací) a také NÚKIB (Národní úřad pro kybernetickou a informační bezpečnost) se sídlem v Brně.

Ke správnému zabezpečení nejenom webového serveru obecně platí neotvírat známé porty směrem ven do veřejné sítě neboli do internetu. Pokud je potřeba z nějakého důvodu port používat, doporučuje se port změnit na jiný, například port 22 pro SSH (Secure Shell) změnit na port 9292.

### 1.5.1 Standard NIST SP 800-52

Institut NIST ve standardu SP 800-52 z roku 2019 poskytuje pokyny pro výběr a konfiguraci implementací protokolu TLS při současném efektivním využívání kryptografických algoritmů. Pro TLS zabezpečení požaduje použití verze TLS 1.2 nebo TLS 1.3. Publikace také poskytuje pokyny k certifikátům [43].

### 1.5.2 Standard NIST SP 800-131A

Standard SP 800-131A z roku 2019 zahrnuje obecný přístup k přechodu z jednoho algoritmu nebo délky klíčů na druhý algoritmus. Standard také poskytuje konkrétnější pokyny k použití robustnějších algoritmů se silnějšími kryptografickými klíči [44].

### 1.5.3 Standard NIST SP 800-57

NIST ve standardu SP 800-57 z roku 2020 publikuje pokyny ke správě kryptografických klíčů. Je rozdělen do tří částí, z nichž první obsahuje obecné pokyny a osvědčené postupy pro správu kryptografických materiálů. Tato část také uvádí definice bezpečnostních služeb, které mohou být poskytovány při použití kryptografie a typů klíčů. Součástí je i specifikace ochrany, kterou každý typ klíče a další kryptografické informace vyžadují, a metody pro zajištění této ochrany s diskuzemi o problémech, které je třeba řešit při použití kryptografie. Druhá část publikace poskytuje pokyny týkající se požadavků na plánování politiky a zabezpečení. Poslední třetí část poskytuje pokyny při používání kryptografických funkcí současných systémů. V následující tabulce 1.1 jsou uvedeny doporučené délky klíče v bitech pro různé algoritmy a délky hašovacích funkcí pro současné systémy [44][45].

Tab. 1.1: Doporučené délky klíčů a hašovacích funkcí podle NIST.

| Rok 2020 | | | | | | |
|---|---|---|---|---|---|---|
| **Síla ochrany** | **Sym. alg.** | **Asym. alg. (modulo)** | Problém diskrétního logaritmu | | **Eliptická křivka** | **Délka haše** |
| | | | **Klíč** | **Grupa** | | |
| 112 | AES-128 | 2048 | 224 | 2048 | 224 | SHA-224<br>SHA-512/224<br>SHA3-224 |
| 128 | AES-128 | 3072 | 256 | 3072 | 256 | SHA-256<br>SHA-512/256<br>SHA3-256 |
| 192 | AES-192 | 7680 | 384 | 7680 | 384 | SHA-384<br>SHA3-384 |
| 256 | AES-256 | 15360 | 512 | 15360 | 512 | SHA-512<br>SHA3-512 |

### 1.5.4 Doporučení na kryptografické algoritmy

Dalším doporučením na kryptografické algoritmy je publikace od úřadu NÚKIB z roku 2018 s názvem Doporučení v oblasti kryptografických prostředků. V něm NÚKIB uvádí, které délky klíče doporučuje k asymetrickým a symetrickým algoritmům, použití módů při šifrování a délky hašovacích funkcí. Také zmiňuje, které algoritmy jsou dosluhující nebo schválené k používání. V tabulce 1.2 jsou znázorněny délky k lepší přehlednosti [46].

Tab. 1.2: Doporučené délky klíčů a hašovacích funkcí podle NÚKIB.

| Rok 2018 | **Délka klíče (výstupu u haše)** | **Algoritmus** |
|---|---|---|
| **Symetrické algoritmy** | 256 | Blokový (AES, Camellia, Serpent) Módy: CCM, GCM, ChaCha20, Poly1305 |
| **Asymetrické algoritmy** | 3072 a více Podgrupa 256 a více | DSA, RSA, DH |
| **Eliptické křivky** | 384 | EC-DSA, EC-Schnorr, ECDH |
| **Hašovací funkce** | 256, 384, 512 | SHA-256, SHA-384, SHA-512, SHA-512/256, SHA3-256, SHA3-384, SHA3-512 |

ENISA ve své publikaci s názvem *Algorithms, key size and parameters* z roku 2014 poskytuje soubor pokynů pro specialisty navrhující a implementující kryptografická řešení pro ochranu osobních údajů v komerčních organizací nebo vládních služeb. Zabývá se postranními kanály, generování náhodných čísel a správě životního cyklu klíčů. V tabulce 1.3 jsou opět uvedeny délky klíčů v bitech, které ENISA doporučuje dodržovat [47].

## 1.6 Nástroje pro testování webových aplikací

V této kapitole budou uvedena současná řešení testování webových aplikací a poté vysvětleny použité nástroje a programy v praktické části této diplomové práce.

Tab. 1.3: Doporučené délky klíčů a hašovacích funkcí podle ENISA.

| Rok 2014 | **Krátkodobé doporučení** | **Dlouhodobé doporučení** |
|---|---|---|
| **Symetrická délka klíče** | 128 | 256 |
| **Výstup hašovací funkce** | 256 | 256, 512 |
| **Faktorizace RSA** | 3072 | 15360 |
| **Problém diskrétního logaritmu** | 3072 | 15360 |
| **Diskrétní logaritmus nad eliptickými křivkami** | 256 | 512 |

### 1.6.1 Webová aplikace SSL Labs

Webová aplikace SSL Labs je nekomerční výzkumný projekt, který obsahuje myšlenky a dokumenty související s SSL. Zároveň poskytuje sbírku nástrojů pro testování zabezpečení SSL/TLS, materiály k pochopení funkčnosti SSL/TLS a různá vylepšení. Zakladatelem projektu SSL Labs je Ivan Ristić.

Aplikace dokáže otestovat bezpečnost vzdáleného webového serveru nebo klientského prohlížeče. Také poskytuje statistiky testování v daném období, detekci zranitelností a doporučení ke zlepšení bezpečnosti [48].

### 1.6.2 Nástroj příkazového řádku testssl.sh

Nástroj **testssl.sh** je bezplatné řešení s otevřeným zdrojovým kódem pro testování vzdáleného serveru. Zjistí zda server podporuje šifrování SSL/TLS, jaké používá protokoly a detekuje známé zranitelnosti serveru. S nástrojem se pracuje v příkazovém řádku a má podporu v Mac OSX a Linux distribucích [49].

### 1.6.3 Projekt OWASP

OWASP (Open Web Application Security Project) je nezisková nadace, která pracuje na zlepšení zabezpečení softwaru. Prostřednictvím softwarů s otevřeným zdrojovým kódem vytvořené komunitami mnoha členů a vzdělávacích konferencí je OWASP zdrojem pro vývojáře k lepšímu zabezpečení webu.

Mezi OWASP projekty patří například **OWASP Top Ten**. Jedná se o standardizovaný dokument o povědomí vývojáře a zabezpečení webových aplikací. Obsahuje

širokou shodu názorů o nejdůležitějších bezpečnostních rizicích pro webové aplikace a používání **OWASP Top Ten** vede organizace k produkci bezpečnějších kódů.

Dalším projektem je například **OWASP Web Security Testing Guide**, který představuje příručku pro testování zabezpečení webu. Slouží pro testování kybernetické bezpečnosti pro vývojáře webových aplikací a bezpečnostní profesionály. Je tvořen společným úsilím profesionálů v oboru kybernetické bezpečnosti a specializovaných dobrovolníků. Obsahuje také rámec osvědčených postupů od testerů penetrace a organizacemi po celém světě [50].

### 1.6.4 Nástroj OpenSSL

OpenSSL je nástroj s otevřeným zdrojovým kódem, který obsahuje kryptografické knihovny a funkce včetně možnosti SSL/TLS implementace. Je napsán v programovacím jazyku C, ale lze ho použít i v jiných jazycích. Umožňuje i práci v příkazovém řádku, kterou lze využít pro správu klíčů, certifikátů a také pro testování různých SSL/TLS spojení [15][51].

#### Tvar šifrovacích sad

Protokoly SSL/TLS používají pro zabezpečené spojení různé šifrovací sady. Sada obsahuje výběr kryptografických primitiv a dalších parametrů potřebných k implementaci zabezpečení. Sada je definována těmito atributy:

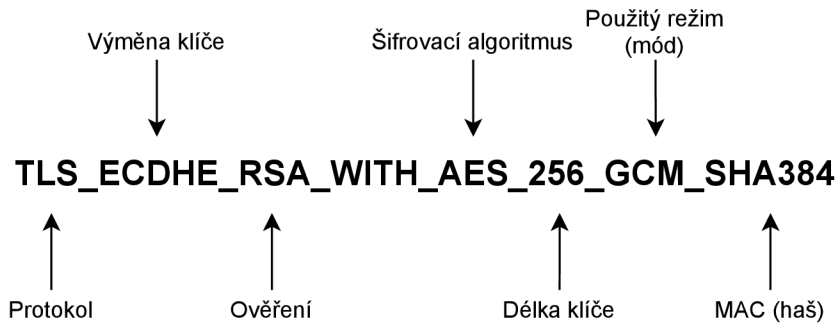
- Metoda ověřování,
- Metoda pro výměnu klíčů,
- Šifrovací algoritmus,
- Velikost šifrovacího klíče,
- Šifrovací režim (pokud je k dispozici),
- MAC algoritmus (pokud je k dispozici).

Název šifrovací sady je složen z názvů použitých atributů, jak je ukázáno na obrázku 1.3 [15]. Ve verzi TSL 1.3 je koncept šifrovací sady změněn a zkrácen [27].

OpenSSL obsahuje ekvivalenty šifrovacích sad, které plní stejnou úlohu, ale některá jejich pojmenování jsou zkrácená. Názvy u protokolu TLS 1.3 jsou totožné. Příklady jsou sepsány v tabulce 1.4 [51].

### 1.6.5 Nástroj Nmap

Nmap (Network Mapper) je bezplatný nástroj s otevřeným zdrojovým kódem pro zjišťování a auditování sítě. Lze monitorovat zařízení nebo služby v síti. Ze zařízení dokáže detekovat operační systémy a jejich verze, názvy služeb a verze aplikací. Byl navržen pro skenování velkých sítí, ale lze ho využít i proti jednotlivým zařízením.




Obr. 1.3: Tvar šifrovací sady.

Tab. 1.4: Ekvivalenty šifrovacích sad s OpenSSL.

| | | |
|---|---|---|
| Příslušná specifikace | TLS_RSA_WITH_AES_128_CBC_SHA256 | TLS 1.2 |
| Ekvivalent v OpenSSL | AES128-SHA256 | TLS 1.2 |
| Příslušná specifikace | TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 | TLS 1.2 |
| Ekvivalent v OpenSSL | ECDHE-RSA-AES256-GCM-SHA384 | TLS 1.2 |
| Příslušná specifikace | TLS_AES_128_GCM_SHA256 | TLS 1.3 |
| Ekvivalent v OpenSSL | TLS_AES_128_GCM_SHA256 | TLS 1.3 |

Je podporován všemi operačními systémy a pro práci je možné používat příkazový řádek nebo grafické rozhraní [52]. V praktické části je použita verze 7.70.

### 1.6.6 Nástroj CURL

Curl je nástroj příkazového řádku a zároveň i knihovnou pro přenos dat pomocí URL adres. Jedná se o software s otevřeným zdrojovým kódem a širokou podporou aplikačních protokolů. Používá se v příkazovém řádku nebo ve skriptech v různých zařízení jako jsou směrovače, televize, tiskárny, mobilní telefony, tablety nebo i v automobilech. Tento nástroj využívají aplikace pro přenos dat na internetu. V současné době ho používá téměř každý člověk využívající internet [53]. V této diplomové práci je použit Curl verze 7.75.0.

## 1.7 Vývojové požadavky pro webovou aplikaci

Obsahem této kapitoly jsou programy a nástroje, které budou použity v této práci. Jedná se o programovací jazyk, skriptovací jazyk a prostředí pro vývoj webové aplikace v praktické části.

### 1.7.1 Programovací jazyk Python

Jazyk Python je univerzální interpretovaný programovací jazyk, který má jednoduchou syntaxi a využívá se pro psaní jasných, logických kódů pro malé i velké projekty. Obsahuje systém shromažďování (garbage-collector), který se stará o automatickou správu paměti a to umožňuje snadnější programování [54].

### 1.7.2 Vývojové prostředí Visual Studio Code

VS Code (Visual Studio Code) je vývojové prostředí s velkou podporou programovacích, skriptovacích a jiných jazyků. VS Code je volně dostupný ke stažení. Jedná se o produkt od společnosti Microsoft a obsahuje spoustu doplňků, které usnadňují programátorovi vývoj. Díky balíčkům byl VS Code vybrán pro praktickou část, kde bude potřeba pracovat s frameworkem Nette a šablonovacím systémem Latte [55].

### 1.7.3 Skriptovací jazyk pro tvorbu webu

Skriptovací jazyk PHP (PHP: Hypertex-tový preprocesor) je velice populární, univerzální a vhodný pro vývoj webových aplikací. Jedná se o jazyk s otevřeným zdrojovým kódem a je bezplatný. Používá se především pro dynamické webové stránky, které PHP vygeneruje pomocí šablon a připravených funkcí. Tyto funkce a šablony stačí být napsané jednou a všude, kde jsou potřeba, se zavolají [56].

## 1.8 Webový servery Apache a Nginx

Webový server se stará o předávání obsahu webové stránky klientovi na jeho žádost. Dále tyto žádosti zpracovává a vrací výsledky zpět klientovi. Jedná se o software, který běží na fyzickém nebo virtuálním serveru a musí zvládat obsluhu více připojených klientů. Nejznámější webové servery jsou Apache a Nginx [57].

### 1.8.1 Apache

Apache HTTP server je vyvíjen s otevřeným zdrojovým kódem s podporou současných moderních operačních systémů včetně Windows a Linux. Cílem Apache je poskytnout bezpečný, efektivní a rozšiřitelný server s aktuálními standardy HTTP [58].

Apache server pro každé spojení s klientem vytváří procesy pro příjem a podprocesy pro zpracování žádostí. U webů s vysokou návštěvností není vhodné tento server aplikovat, jelikož tyto procesy a podprocesy vytěžují hardwarové prostředky serveru. Apache dokáže zpracovávat dynamický obsah, který využívá např. skriptovací jazyk PHP, aniž by se musel předávat externí komponentě. Dále při vkládání dalších modulů například pro přepisování URL adres do Apache serveru není potřeba restart a jednotlivé složky lze konfigurovat pomocí souboru *.htaccess* [57].

### 1.8.2 Nginx

Webový server Nginx vznikl v roce 2004. Také se jedná o software s otevřeným zdrojovým kódem a součástí Nginx je nejen HTTP server, ale i mailový server a další služby [59].

Umožňuje zpracovávat žádosti od klientů efektivněji oproti Apache serveru pomocí událostmi řízené architektury. Díky tomu jeden proces dokáže zpracovávat tisíce nových spojení. Nginx pracuje se statickým obsahem, tudíž pro dynamickou práci je potřeba externě přidat komponentu, která se o dynamický obsah postará.

Nginx je tedy vhodný použít pro web se statickým obsahem a vysokou návštěvností [57].

### 1.8.3 Porovnání webových serverů

Níže v tabulce 1.5 je vypsán souhrn vlastností zmíněných webových serverů [57].

Tab. 1.5: Porovnání webových serverů Apache a Nginx.

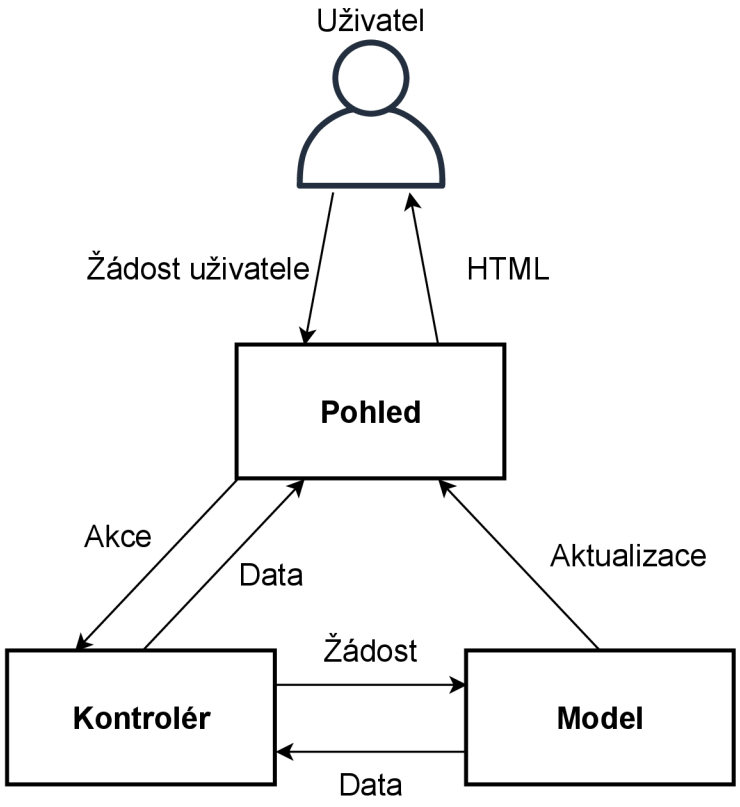
| **Apache** | **Nginx** |
|---|---|
| Dynamické zpracování obsahu | Statické zpracování obsahu |
| Vytváří procesy a podprocesy, které vytěžují server | Vytvoří jeden proces |
| Dynamické vkládání modulů | Práce s moduly je komplikovanější |
| Otevřený zdrojový kód | Otevřený zdrojový kód |
| Podpora všech operačních systémů | Podpora operačních systémů s omezením |

V praktické části této práce byl vybrán webový server Apache a to díky flexibilitě modulů a práci s dynamickým obsahem.

## 1.9 Architektura Model-View-Controller

MVC (Model-View-Controller) architektura představuje vzor, podle kterého se oddělují tři hlavní části aplikace. Každá část vykonává specifické aspekty vývoje aplikace. MVC je nejpoužívanější způsob vývoje webových aplikací.

Model představuje logickou část aplikace, která pracuje s daty a databázemi. Pohled (View) se používá pro výstup do uživatelského rozhraní aplikace. Zahrnuje textová pole nebo rozbalovací nabídky, s kterými uživatel interaguje. Kontrolér (Controller - Řadič) funguje jako rozhraní mezi komponentami Model a Pohled. Od uživatele zpracovává příchozí požadavky a manipuluje s daty pomocí Modelu. Následně přes Pohled vykreslí výstupy. Schéma modelu je znázorněno na obrázku 1.4 [60].




Obr. 1.4: Architektura MVC.

### 1.9.1 Framework Nette 3

Framework Nette je rámec pro tvorbu webové aplikace, obsahuje komponenty a ulehčuje práci při tvorbě webu, který je stavěn na PHP jazyku. Taktéž obsahuje vlastní šablonovací systém *latte* a ladící nástroje.

Nette vychází z architektury MVC, ale místo kontroléru používá Presentery tedy MVP [61].

### 1.9.2 Výhody a nevýhody

Výhodou Nette je česká dokumentace a podpora. Dále je vyvíjen s otevřeným zdrojovým kódem, takže je velice modulární. Zpracovaná dokumentace je velice přehledná a obsahuje příklady použití. Také disponuje jednoduchým šablonovacím systémem.

Jednu nevýhodu lze nalézt v pevně daném stylu vývoje, ale to platí pro framework programy obecně.

# 2 Praktická část

Praktická část v této diplomové práce se zabývá vývojem webové aplikace ve frameworku Nette, která bude testovat použité kryptografické sady TLS protokolu na vzdáleném serveru. Připraví se prostředí pro vývoj a bude vytvořen návrh aplikace s využitím automatických skriptů a následné grafické zobrazení výsledků zjištěných informací.

## 2.1 Příprava prostředí

Jako vývojové prostředí se použije Visual Studio Code na virtuálním počítači Debian s nainstalovanými rozšířeními Python a Nette Latte + Neon. Bude lepší vyvíjet aplikaci přímo v Debianu, aby se ulehčila práce s případným laděním a vyhnutí se neustálým nahráváním nových úprav na webový server. Distribuční verze Debianu je 10.6. Dalé v něm běží webový server Apache verze 2.4.38.

### 2.1.1 Instalace VSCode na Debian

Je potřeba stáhnout balíček **.deb** vývojového prostředí VSCode ze stránky
*https://code.visualstudio.com/Download*

Pro instalaci VSCode se použije příkaz
*sudo apt install code_1.50.0-1602051089_amd64.deb*

Následně se spustí příkazem
*code*

### 2.1.2 Nastavení Apache serveru a instalace PHP

V Apache serveru se vytvoří virtuální host v adresáři **/var/www** a pojmenuje se podle potřeby například **appserver.test**
*sudo mkdir -p /var/www/appserver.test/html*

Tento název bude použit v práci jako název webové aplikace. Dále je nutné vytvořit konfigurační soubor pro virtuálního hosta
*sudo nano /etc/apache2/sites-available/appserver.test.conf*

Do konfiguračního souboru se doplní nastavení podle výpisu 2.1. Toto nastavení poslouží pro budoucí potřeby při vývoji webové aplikace.

Vytvořený virtuální server neobsahuje skriptovací jazyk PHP. Ten je důležitý pro běh celé aplikace. Instalace se provede postupným provedením příkazů níže. Přidáme nový repozitář, ze kterého se stáhne a nainstaluje PHP, a aby se předešlo problémům, nainstalují se i doplňující součásti [62]
*sudo apt -y install lsb-release apt-transport-https ca-certificates*
*sudo wget -O /etc/apt/trusted.gpg.d/php.gpg https://packages.sury.org/php/apt.gpg*

*echo "deb https://packages.sury.org/php/ $(lsb_release -sc) main" | sudo tee /etc /apt/sources.list.d/php.list*

Poté se aktualizují balíčky ze seznamu repozitářů
*sudo apt update*

Následně samotná instalace PHP verze 7.4
*sudo apt -y install php7.4*

Instalace doplňujících součástí PHP
*sudo apt-get install php7.4-{bcmath,bz2,intl,gd,mbstring,mysql,zip}*

Výpis 2.1: Konfigurační soubor virtuálního hosta.

```
<VirtualHost *:80>
    ServerAdmin admin@appserver.test
    ServerName appserver.test
    ServerAlias www.appserver.test
    DocumentRoot /var/www/appserver.test/html/nette-app/www
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>

<Directory "/var/www/appserver.test/html/nette-app/www">
    AllowOverride All
</Directory>
```

### 2.1.3 Instalace Composeru a implementace Nette frameworku

Composer je správce závislostí pro PHP a pomocí něho se stáhne základní webová aplikace obsahující Nette framework. Instalační soubor správce se stáhne přes PHP skript
*php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"*

Ověří se stažený správce pomocí haše. Tato kontrola haše je závislá na konkrétní verzi Composeru, proto je důležité postupovat podle návodu vývojáře [63]
*php -r "if (hash_file('sha384', 'composer-setup.php') === '795f976fe0ebd8b75f26 a6dd68f78fd3453ce79f32ecb33e7fd087d39bfeb978342fb73ac986cd4f54edd0dc902601 dc') { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"*

Dále se spustí instalátor s parametry –**install-dir**, který definuje místo své instalace, a –**filename**, který přejmenuje spustitelný skript
*php composer-setup.php –install-dir=bin –filename=composer*

Nakonec se smaže zbytečný instalátor
*php -r "unlink('composer-setup.php');"*

#### Příprava Nette prostředí

Nyní je Composer připraven k použití. Do adresáře **/var/www/appserver.test/html** se stáhne základní webová aplikace s názvem **nette-app** příkazem

*composer create-project nette/web-project nette-app*

Nakonec se přidělí práva zápisu do složek **temp** a **log** [64] a aktivuje se virtuální host
*cd /var/www/appserver.test/html/nette-app*
*chmod -R a+rw temp log*
*sudo a2ensite appserver.test*
*sudo systemctl restart apache2*

Po stažení je webová aplikace dostupná na adrese `http://www.appserver.test/` jenom za předpokladu, že je v souboru *hosts* vložen záznam pro překlad adresy.

### 2.1.4 Popis adresářové struktury frameworku Nette

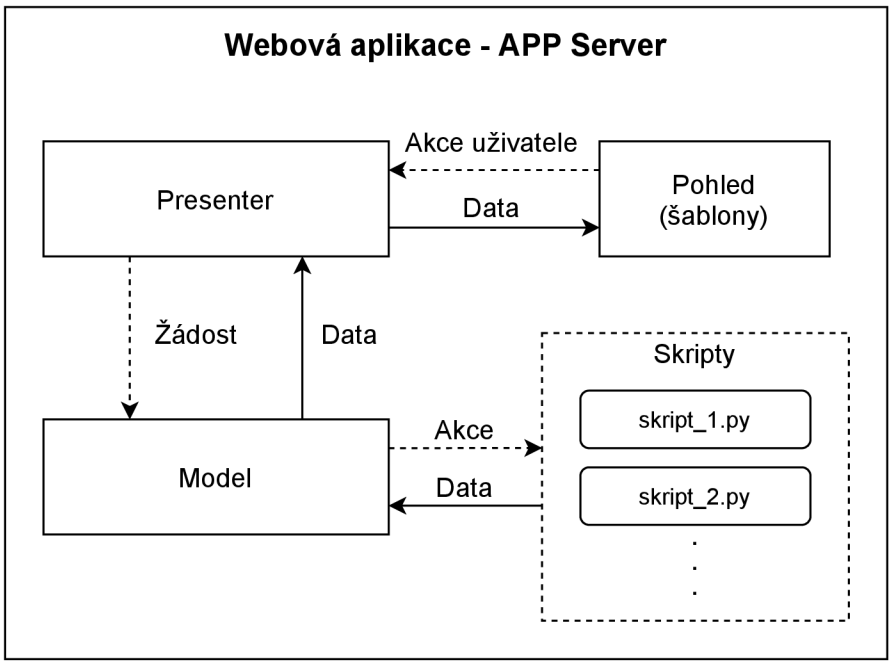
Struktura stažené základní aplikace obsahuje [65]:

- `nette-app/` ..... kořenový adresář aplikace
  - `app/` ..... adresář s aplikací
    - `config/` ..... konfigurační soubory
    - `Presenters/` ..... třídy presenterů
      - `templates/` ..... základní šablony
    - `Router/` ..... konfigurace URL adres

```
    └── Bootstrap.php ..... zaváděcí třída Bootstrap
├── bin/ ..... skripty spouštěné z příkazové řádky
├── log/ ..... logování chyb
├── temp/ ..... dočasné soubory, cache, a další
├── vendor/ ..... knihovny instalované Composerem
│   └── autoload.php ..... automatické načtení všech nainstalovaných balíčků
└── www/ ..... veřejný webový adresář
    └── index.php ..... soubor, kde se spouští celá aplikace
```

## 2.2 Návrh webové aplikace

Schéma návrhu webové aplikace vychází z modelu MVC. Na obrázku 2.1 je popsán způsob získávání dat pomocí skriptů, jejich zpracování a zobrazení na webové stránce uživateli. Model spouští testovací skripty na vzdálený server a výsledky předá presenteru. Skripty se spouštějí postupně a jsou zaměřené na konkrétní účel zjištění. Například pro detekci zapnutého mechanismu HSTS na vzdáleném serveru.

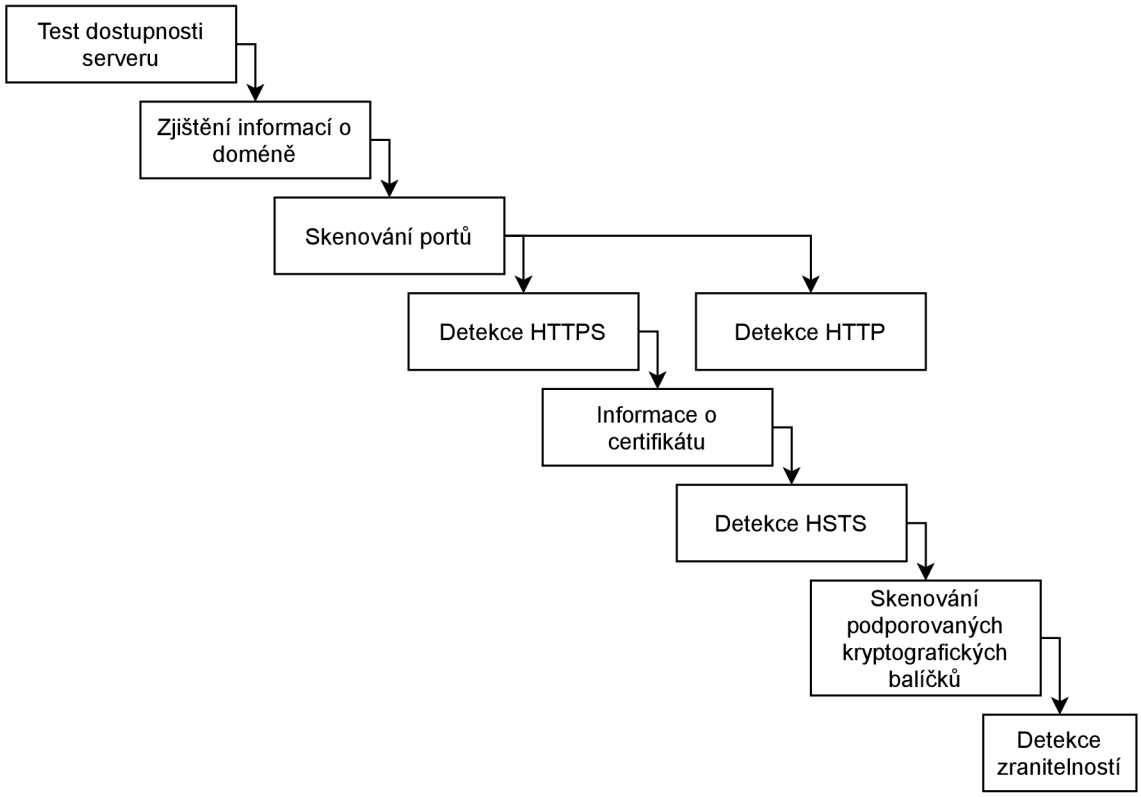



Obr. 2.1: Schéma návrhu webové aplikace.

### 2.2.1 Postup testování vzdáleného serveru

Postup, jakým se bude vzdálený server testovat, je zobrazen na obrázku 2.2. V první řadě se zjistí dostupnost vzdáleného serveru a informace o vzdáleném serveru například IP (Internet Protocol) adresa. Poté se oskenují porty, aby se vědělo, které

jsou otevřené a mohou být potenciální hrozbou v bezpečnosti. Na základě zjištěných portů se detekuje HTTPS na portu 443. Pokud ho vzdálený server nemá otevřený, nelze testovat bezpečnost HTTPS a je možné, že podporuje pouze nezabezpečený HTTP. Dále se zjistí informace o certifikátu v případě detekce HTTPS a následně se testuje mechanismus HSTS, který zabraňuje degradaci HTTPS spojení na HTTP. V posledních krocích je skenování podporovaných kryptografických balíčků na vzdáleném serveru a na základě toho se detekují zranitelnosti, kterými je server vystaven.




Obr. 2.2: Postup testování vzdáleného serveru.

## 2.3 Zprovoznění aplikace

V této kapitole se vytvoří modul aplikace a první nastavení pro budoucí grafické úpravy. Modul představuje logickou část aplikace, která může zahrnovat několik presenterů, šablon a modelů. Také se vytvoří první skript, který vyčte ze zadané webové stránky seznam podporujících sad šifrovacích algoritmů pro jednotlivé verze SSL/TLS protokolů.

### 2.3.1 Webová část aplikace

Vlastní modul do frameworku Nette se implementuje vytvořením složky **CoreModule** v adresáři **app**. Modul se může pojmenovat jakkoliv, ale je důležité dodržet formát nazvu ve tvaru **<název>Module**.

Do vytvořeného modulu se vytvoří potřebné adresáře **Presenters**, **models**, **config** a **templates**. Ve složce **temlates** bude umístěna šablona pro presenter s názvem stejným jako obsahuje presenter. Nyní vypadá adresářová struktura modulu následovně:

```
app/ ..... adresář aplikace
├── ...
├── CoreModule/
│   ├── config/
│   ├── models/
│   ├── Presenters/
│   └── templates/
└── ...
```

Vytvoří se jednoduchý model v adresáři **CoreModule/models**, který bude zatím předávat data z presenteru a poté mu ta samá data vrátí. Takto se odzkouší, zda model funguje, jak má. Nazve se například *ProtocolsCiphersManager.php*. Opět se musí dodržet formát názvu pro model ve tvaru **<název>Manager**. Vloží se zde kód podle výpisu 2.2.

Výpis 2.2: Model ProtocolsCiphersManager.php v modulu CoreModule.

```
<?php

namespace App\CoreModule\Models;

use Nette;
use Nette\SmartObject;

class ProtocolsCiphersManager {
    public function runTest($url): string {
        return $url;
    }
}
```

V adresáři **CoreModule/config** se vytvoří konfigurační soubor *config.neon* a do něho se přidá služba pro CoreModule podle výpisu 2.3.

Výpis 2.3: Konfigurační soubor config.neon v modulu CoreModule.

```
services:
    - App\CoreModule\Models\ProtocolsCiphersManager
```

Výpis 2.4: Hlavní konfigurační soubor common.neon.

```
# Propojení s dalšími konfiguračními soubory.
includes:
    # Načtení konfigurace z CoreModule.
    - ../CoreModule/config/config.neon
```

Konfigurační soubor modulu se musí přiřadit podle výpisu 2.4 k hlavnímu konfiguračnímu souboru *common.neon* v adresáři **app/config/**.

Do adresáře **CoreModule/Presenters** se vytvoří presenter s názvem ve tvaru *<název>Presenter.php*, tedy stejný proměnný název, jako obsahuje model. Výpis souboru *ProtocolsCiphersPresenter.php* je uveden v příloze C.1. Metoda v presenteru *createComponentServerForm()* vytvoří formulář, který se zobrazí na webové stránce. Tato komponenta má také důležitý formát názvu v metodě ve tvaru **createComponent<názevKomponenty>**. Díky tomu lze v šabloně vypsat formulář podle klíčového názvu komponenty **<názevKomponenty>**. Po úspěšném odeslání formuláře ze stránky se spustí metoda *formSucceeded(Form $form, $data)* se vstupními argumenty. Argument **form** odkazuje na daný formulář a argument **data** obsahuje jednotlivé vstupy z formuláře. Na konkrétní vstupy se odkazuje ve tvaru *$data->server*. Tato metoda předá potřebné informace pro model.

Po vykonání akcí v modelu se výsledek uloží zpět do presenteru a metodou *renderDefault()* předá proměnné do šablony. V šabloně se proměnné vypíšou podle stejného názvu. Ukázka ve výpisu 2.5.

Výpis 2.5: Metoda renderDefault() a proměnné v šabloně.

```
# Presenter
...
public function renderDefault(): void {
        $this->template->output = $this->output;
}
...

# Šablona
<p>{$output}</p>
```

Šablona pro první spuštění aplikace je uložena v adresáři **CoreModule/templates**, kde se vytvoří složka **ProtocolsCiphers** se stejným názvem jako obsahuje presenter. Do této složky se vytvoří základní šablona pro presenter *default.latte*. Obsah šablony je znázorněn ve výpisu 2.6.

Výpis 2.6: Šablona default.latte pro presenter ProtocolsCipherPresenter.php.

```
<title>{block title}APP Server{/block}</title>
{block styles}
  <link rel="stylesheet"
  href="{$basePath}/css/bootstrap.min.css">
{/block}

{block content}

<div class="container">
    <div class="col-md-6␣mx-auto">
    {control serverForm}
    </div>
    <div class="col-md-6␣mx-auto">
    <p>{$output}</p>
    </div>
</div>
{/block}
```

Všechny šablony pro jednotlivé presentery se dosazují do hlavní šablony *layout.latte* v adresáři **app/Presenters/templates**. Do ní se vloží skripty pro framework Bootstrap, který se bude starat o grafickou stránku webu a do hlavičky se přidá *include* pro styly v šabloně **ProtocolsCiphers**. Ukázka ve výpisu A.1.

Každý vytvořený presenter musí mít vytvořenou cestu pro obousměrné překládání mezi URL a akcí presenteru. Toto se nastavuje v *RouterFactory.php* v adresáři **app/Router**. Zde se změní cesta v metodě *createRouter()* podle výpisu 2.7.

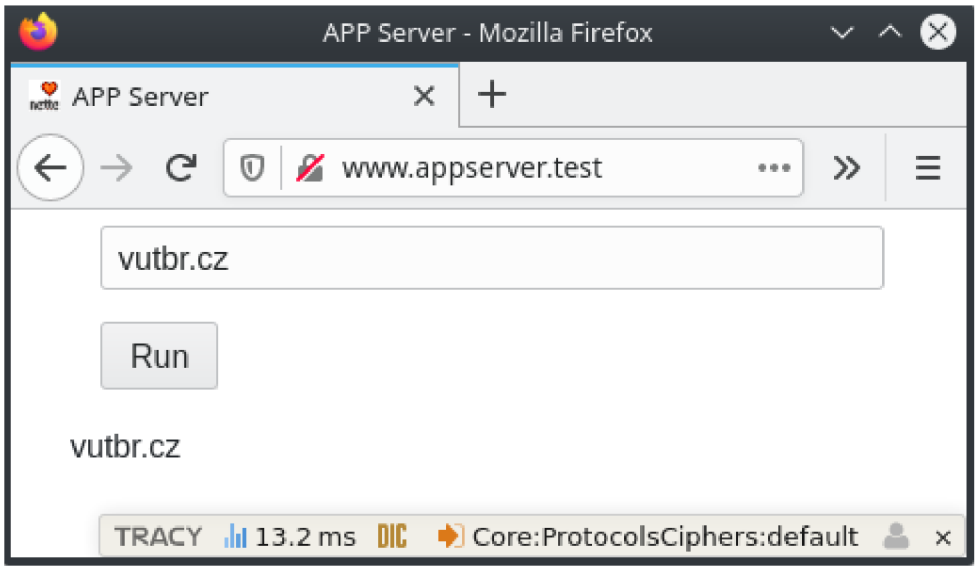
Výpis 2.7: Nastavení cesty pro presenter v RouterFactory.php.

```
...
    $router = new RouteList;
    $router->addRoute('<presenter>/<action>[/<id>]',
'Core:ProtocolsCiphers:default');
    return $router;
...
```

Po aktualizaci webové aplikace na adrese `http://www.appserver.test/` se zobrazí formulář jako na obrázku 2.3 pro zadání vzdáleného serveru. Po kliknutí na tlačítko **Run** se zatím zobrazí pouze zadaný server. Tímto se ověřila funkčnost webové aplikace.

Někdy je potřeba pro správnou funkčnost aplikace a zobrazení stránky smazat

dočasně uložené soubory příkazem
*sudo rm -R /var/www/appserver.test/html/nette-app/temp/cache/*




Obr. 2.3: Ukázka spuštěné webové aplikace.

### 2.3.2 Vytvoření prvního skriptu

První skript, který byl vytvořen, využívá nástroj **OpenSSL** verze 1.1.1d. Ukázka napsaného skriptu je v příloze B.1. Skript je napsán v pythonu. Hlavní funkce *main()* zpracuje vstupní argument a předá ho do funkce *test_protocols()*. Argumentem v tomto případě je testovaný server ve tvaru *example.com:443*. Tato funkce spustí složený příkaz nástroje Openssl a výstup zaznamená do proměnné *suits*. Proměnná *suits* se vypíše, a tím se výstup zaznamená do modelu **ProtocolsCiphersManager** ve webové aplikaci. Příkaz testuje SSL/TLS spojení s různými variantami kryptografických balíčků, které OpenSSL podporuje. Pokud se spojení naváže, zaznamená se protokol a použitý kryptografický balíček algoritmů ve tvaru <PROTOKOL>:<CIPHERSUIT>. Veškeré skripty se budou ukládat do adresáře **bin**.

## 2.4 Zobrazení výsledků skriptu v aplikaci

V modelu **ProtocolsCiphersManager** se upraví metoda *runTest()* podle výpisu 2.8. Pro ověření funkčnosti je vytvořena proměnná *test*, která odpovídá výstupu skriptu. Je vhodné pro toto ověření nepoužívat vytvořený funkční skript, jelikož jeho výstup nějakou dobu trvá. Dále tato metoda volá metodu *fill_protocols_and_ciphers_to_arrays()*. Pro otestování funkčního skriptu se odkomentují proměnné

Výpis 2.8: Metoda runTest() pro první skript.

```
...
public function runTest($url): void {
        #$command = escapeshellcmd('../bin/openssl.py␣'.$url);
        $command_openssl = 'openssl␣version';
        #$out = shell_exec($command);
        #$out_new = str_replace("\n", '', $out);
        $this->openssl_version = shell_exec($command_openssl);
        $test = "tls1_1:ECDHE-RSA-AES256-SHA,
tls1_1:ECDHE-RSA-AES128-SHA,
tls1_2:ECDHE-RSA-AES256-GCM-SHA384,
tls1_3:ECDHE-RSA-AES256-GCM-SHA384,
tls1_3:ECDHE-RSA-AES128-SHA,";
        $this->fill_protocols_and_ciphers_to_arrays($test);
    }
...
```

*command*, *out*, *out_new*, zakomentuje se *test* a změní se vstupní parametr metody *fill_protocols_and_ciphers_to_arrays()* na *$out_new*.

Výsledek skriptu vstupuje do metody *fill_protocols_and_ciphers_to_arrays()*, která roztřídí a vloží zjištěné informace do proměnných *protocols*, *prot_count* a *arrayP*. Ukázka ve výpisu C.4.

V presenteru se upraví metoda *renderDefault()*, která se volá vždy po ukončení daného presenteru. Podle výpisu C.2 se předají hodnoty, se kterými se bude pracovat v šabloně. Dále se vytvoří pomocná proměnná *success*, která se nastaví na TRUE, pokud se úspěšně potvrdí formulář v metodě *formSucceeded()*, a pomocný slovník protokolů *dic_prot*, který převede protokoly na vzhledově lepší řetězec. Toto je zobrazeno ve výpisu C.3.

## 2.5 Grafická úprava webového prostředí

Do šablony *default.latte* modulu **Core** se připraví HTML struktura podle výpisu D.1 pro zobrazení tabulek, do kterých se pomocí cyklů **foreach** vloží konkrétní hodnoty. Proměnná *success* ošetřuje zobrazení tabulek, až po úspěšném vykonání spuštěného skriptu tlačítkem. Pro využití vizuálního CSS stylu z frameworku Bootstrap 4 je tagům nutné správně přiřadit třídy [66] [67].

### 2.5.1 Přidání pravidla pro vstupní formulář

Do formuláře se musí zadat validní adresa s portem pro HTTPS. Proto je důležité, aby se kontroloval vstup. Pro tento účel je vytvořen *InputValidator.php* v adresáři modulu **CoreModule**. Ten bude kontrolovat základní tvar vstupního řetězce ve formuláři ve tvaru *example.com:443*. Validátor je ukázán ve výpisu 2.9. Dále se do formuláře pro zadání textu přiřadí pravidlo *addRule()*, které zavolá metodu *isCorrect()* ve validátoru. Pravidlo se aplikuje podle výpisu 2.10.

Výpis 2.9: Kontrola vstupu pomocí InputValidator.php.

```
class InputValidator
{
  // Otestuje zda je správně zadán vstup
  public static function isCorrect($input, $arg): bool
  {
    if(strpos($input->getvalue(), ":") !== false) {
      $s = explode(':', $input->getvalue());
      if ($s[0] !== "" && $s[1] !== "") {
        if ($s[1] == 443) return true;
        else return false;
      } else return false;s
    }
    else return false;
  }
}
```

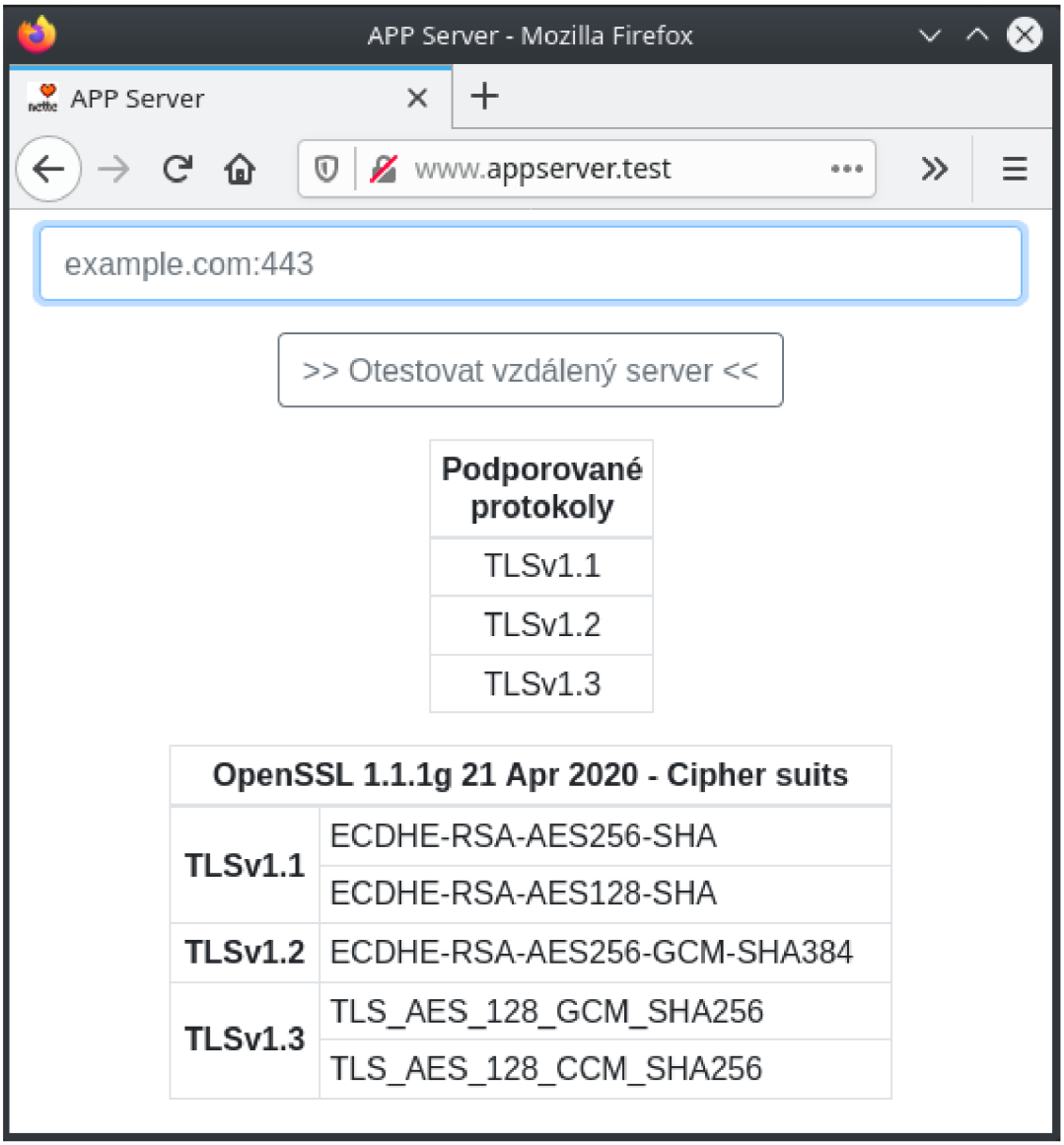
Výpis 2.10: Přiřazení pravidla do vstupního formuláře.

```
use App\CoreModule\InputValidator;
class ProtocolsCiphersPresenter extends Presenter {
  protected function createComponentServerForm(): Form {
    ...
    $form->addText('server', '')
    ->setRequired()
    ->setHTMLAttribute('class', 'form-control')
    ->setHTMLAttribute('placeholder', 'example.com:443')
    ->addRule(
      'InputValidator::isCorrect',
      'Zadaný␣server␣musí␣být␣ve␣formátu:␣example.com:%d',
      443);
    $form
    ->addSubmit('send', '>>␣Otestovat␣vzdálený␣server␣<<')
    ->setHTMLAttribute('class', 'btn␣btn-outline-secondary');
```

```
        $form->onSuccess[] = [$this, 'formSucceeded'];
        return $form;
    }
}
```

Obrázek 2.4 znázorňuje grafickou podobu současné webové aplikace po výpisu výsledků testování vzdáleného serveru. V horní části se nachází vstupní textové pole pro zadání vzdáleného serveru a pod ním tlačítko, které test spustí. Načítání stránky bude trvat v závislosti na počtu podporovaných protokolů s šifrovacími balíčky. Po dokončení testu se výsledky zobrazí v tabulkách, jak je vidět na obrázku 2.4.

V příloze E.1 je ukázka výsledku prvního skriptu při testování webového serveru *vutbr.cz*.




Obr. 2.4: Grafická podoba webu po výstupu prvního skriptu.

## 2.6 Implementace dalších skriptů

V následujících podkapitolách budou vysvětleny jednotlivé postupy testování na obrázku 2.2, vytvořené skripty a úpravy kódu ve webové aplikaci **appserver.test**. Skripty jsou vytvořené tak, aby mohly být použité i v jiných aplikacích a nejsou nijak vázané na framework Nette.

Jelikož postup testování zahrnuje detekci HTTPS, není potřeba zadávat adresu s portem 443 do vstupního formuláře jako to bylo v kapitole 2.5.1. Také se upraví validátor *InputValidator.php* podle výpisu 2.11.

Výpis 2.11: Výpis úpravy validátoru.

```
public static function isCorrect($input, $arg): bool
{
  if(strpos($input->getvalue(), ".") !== false) {
    $s = explode('.', $input->getvalue());
    if ($s[0] !== "" && $s[1] !== "") {
      return true;
    } else return false;
  }
  else return false;
}
```

### 2.6.1 Test dostupnosti vzdáleného serveru

Na začátku celého testování je potřeba ověřit, zda je vzdálený webový server dostupný. Jednoduchou metodou, jak tohoto docílit, je spustit nástroj **ping**. Pokud **ping** dorazí na cílový server a přijde zpět odpověď **Reply**, může se pokračovat v testu. V případě nedostupnosti serveru nebude test spuštěn. Vzdálený webový server buď neexistuje nebo má zakázaný ICMP (Internet Control Message Protocol) protokol.

#### Python skript

Ve výpisu 2.12 je ukázka skriptu, který spustí ping s parametrem **-c**, který znamená v linuxových systémech počet poslaných žádostí. Dále se zadá cílový server a řetězec *< /dev/null > /dev/null 2>&1 && echo ONLINE*, který v případě dostupnosti serveru vypíše řetězec ONLINE a při nedostupnosti serveru nevypíše nic.

Výpis 2.12: Výpis skriptu pingtest.py.

```
#!/usr/bin/env python3
import os, argparse, io
```

```
def ping_test(server):
    out = ''
    # Testuje se, zda je server dosažitelný, pokud ano,
    # vypíše se řetězec ONLINE
    out += os.popen('ping -c 3 '+server+
    ' < /dev/null > /dev/null 2>&1 && echo ONLINE').read()
    print(out)

def main():
    parser = argparse.ArgumentParser("Need target")
    parser.add_argument("server", type=str)
    args = parser.parse_args()
    ping_test(args.server)

if __name__ == '__main__':
    main()
```

### Metoda v modelu ProtocolsCiphersManager

Do modelu **ProtocolsCiphersManager** se vytvoří metoda *runPing()*, která spustí skript a na základě výstupu nastaví proměnnou *isOnline*. Metodu lze vidět ve výpisu 2.13.

Výpis 2.13: Výpis metody runPing().

```
private function runPing(): bool {
    $command = escapeshellcmd('../bin/pingtest.py '.$this->
        url);
    $out = shell_exec($command);
    $out_new = str_replace("\n", '', $out);
    if ($out_new == "ONLINE") {
        $this->isOnline = TRUE;
        return $this->isOnline;
    }
    else return $this->isOnline = FALSE;
}
```

### Předání proměnných v presenteru ProtocolsCiphersPresenter

V metodě *renderDefault()* v presenteru **ProtocolsCiphersPresenter** se předají proměnné z modulu do šablony tak, jak je to znázorněno ve výpisu 2.14.

#### Úprava v šabloně default.latte

V šabloně *default.latte* se vytvoří podmínka. Pokud bude server dostupný, vypíše se blok obsahující HTML tagy. V případě nedostupnosti serveru se vypíše text, který je vidět ve výpisu 2.15.

Výpis 2.14: Výpis předání proměnné do šablony.

```
public function renderDefault(): void {
    ...
    $this->template->online = $this->protocolsCiphersManager->
        isOnline;
}
```

Výpis 2.15: Základní HTML struktura.

```
<div class="container">
  ...
  {if $success}
  {if !$online}
    <p style="text-align:center;">Server nenalezen. Zkuste to
    znovu nebo později.</p>
  {else}
  {* HTML obsah v případě dostupnosti vzdáleného webového
  serveru *}
  {/if}
  {/if}
</div>
```

### 2.6.2 Zjištění informací o doméně

Při dostupnosti webového serveru se pokračuje s dalším testováním. Ve skriptu *nmapinfo.py* v příloze F.1 je implementovaný příkaz nástroje **Nmap**, který parametrem **-p** otestuje vybrané porty, zda jsou otevřené. Současně se spustí v příkazu skript **whois-ip** parametrem **--script**, který zjistí informace o doméně. Tvar výstupu je následující:
*<doménové jméno a IP adresa>*
*<alternativní IP adresy>*
*<seznam oskenovaných portů oddělené středníkem>*
*<informace o poskytovateli oddělené středníkem>*

### 2.6.3 Skenování portů

Skriptem *nmapinfo.py* se tedy zjistí i stavy oskenovaných portů. To může být z hlediska bezpečnosti také důležité, protože mít zbytečně otevřené porty útočníky láká hádat hesla např u FTP (File Transfer Protocol) protokolu na portu 21. V tabulce 2.1 jsou vypsané porty s popisem, které se testují ve skriptu v příloze F.1.

Tab. 2.1: Seznam skenovaných portů.

| Port | Popis |
|---|---|
| 21 | FTP protokol pro přenos souborů |
| 22 | SSH zabezpečený komunikační protokol v síti |
| 23 | Telnet nezabezpečený komunikační protokol v síti |
| 53 | DNS komunikace pro překlad doménových jmen na IP adresy |
| 80 | HTTP nezabezpečená komunikace s webovým serverem |
| 113 | Ident protokol k ověření identifikace uživatele v síti |
| 443 | HTTPS zabezpečená komunikace s webovým serverem |
| 3389 | RDP vzdálené ovládání počítače |

#### Metody v modelu ProtocolsCiphersManager

V modelu **ProtocolsCiphersManager** se vytvoří metoda *runInfoScan()*, která spustí skript *nmapinfo.py*. Tu lze vidět ve výpisu 2.16. Následně se v podmínce zavolá metoda *get_info_about_server()* se vstupní proměnnou *out*, která je výstupem skriptu. Metoda *get_info_about_server()* rozdělí zjištěné informace a naplní je do proměnných k nim určené. Celý výpis metody *get_info_about_server()* je v příloze F.3.

Výpis 2.16: Výpis metody runInfoScan().

```
private function runInfoScan(): bool {
    $command = escapeshellcmd('../bin/nmapinfo.py␣'.$this->
        url);
    $out = shell_exec($command);
    if($this->get_info_about_server($out))
        return true;
    else return false;
}
```

#### Předání proměnných v presenteru ProtocolsCiphersPresenter

Tak jako ve výpisu 2.14 se předají všechny proměnné z modelu do šablony *default.latte*.

### 2.6.4 Detekce HTTP/HTTPS

Výstupem metody *get_info_about_server()* je **Pravda** při detekci otevřeného portu 443 HTTPS a **Nepravda** při neotevřeném portu. Také se zjistí stav portu 80 HTTP. Na základě kladné detekce HTTPS se pokračuje v testování vzdáleného serveru. Podmínky testování stavů portů jsou ukázány v příloze F.3.

### 2.6.5 Zjištění informací o certifikátu

Po úspěšné detekci HTTPS je nyní potřeba zjistit informace o certifikátu vzdáleného serveru. K určení bezpečnosti serveru může být důležitá například platnost certifikátu nebo použitá délka veřejného klíče. Ve skriptu *certinfo.py* v příloze F.2 jsou použity nástroje **OpenSSL** a **Nmap**. Zjišťují se následující informace v tomto tvaru:
*<informace o subjektu>*
*<informace o vydavateli>*
*<podpora znovu vyjednávání>*
*<používaná komprese>*
*<typ veřejného klíče>*
*<délka veřejného klíče>*
*<algoritmus podpisu>*
*<platnost certifikátu OD>*
*<platnost certifikátu DO>*

Prvním příkazem nástrojem **OpenSSL** se parametrem **-connect** naváže spojení se vzdáleným serverem na portu 443 a zjistí se informace o subjektu, vydavateli, podpoře opětovného vyjednávání a používané komprese. Ve druhém příkazu se s nástrojem **Nmap** zjistí typ a délka veřejného klíče, použitý algoritmus k podpisu a začátek a konec platnosti certifikátu. K získání těchto informací se použije vestavěný skript **ssl-cert** parametrem **--script** v nástroji **Nmap**. Nakonec parametrem **-p** definujeme port HTTPS protokolu.

#### Metody v modelu ProtocolsCiphersManager

Opět se v modelu **ProtocolsCiphersManager** vytvoří nová metoda *runCertInfo()*, která spustí skript *certinfo.py* a výstup skriptu předá do metody *get_cert_info()*

Výpis 2.17: Výpis metody runCertInfo().

```
private function runCertInfo(): void {
    $command = escapeshellcmd('../bin/certinfo.py␣'.$this->
url);
    $out = shell_exec($command);
    $this->get_cert_info($out);
}
```

argumentem *out*. Metodu *runCertInfo()* lze vidět ve výpisu 2.17.

Metoda *get_cert_info()* v příloze F.4 vstupní data vhodně rozdělí a uloží do proměnných. Také provede kontrolu platnosti certifikátu.

#### Předání proměnných v presenteru ProtocolsCiphersPresenter

Všechny používané proměnné v metodě *get_cert_info()*, které jsou použité pro uložení informací o certifikátu se předají v presenteru **ProtocolsCiphersPresenter** do šablony podobně jako ve výpisu 2.14.

### 2.6.6 Detekce HSTS

K určení podpory HSTS mechanismu se použije nástroj **Curl**. Ve výpisu 2.18 lze vidět příkaz s parametry. Parametr **-s** zapne tichý režim bez znázornění ukazatele průběhu. Parametr **-D** uloží HTTP hlavičku, která byla poslána od serveru a pomlčkou **-** se převedou data do standardního výstupu **stdout**. Nakonec se z uložené hlavičky vytáhne řetězec *strict* příkazem **grep**, jenom pokud webový server HSTS podporuje.

Výpis 2.18: Výpis skriptu hststest.py.

```
#!/usr/bin/env python3
import os, argparse, io

def hsts_test(server):
    out = ''
    hsts = ''
    # Testuje zda HTTPS hlavička obsahuje zabezpečení HSTS
    out += os.popen('curl␣-s␣-D␣-␣https://'+server+'/␣\
|␣grep␣-i␣strict').read()
    bf = io.StringIO(out)
    line = bf.readline()
    if "strict" in line:
```

```
        hsts = "HSTS"
    print(hsts)

def main():
    parser = argparse.ArgumentParser("Need target")
    parser.add_argument("server", type=str)
    args = parser.parse_args()
    hsts_test(args.server)

if __name__ == '__main__':
    main()
```

### Metoda v modelu ProtocolsCiphersManager

V modelu **ProtocolsCiphersManager** se vytvoří další metoda s názvem *runHstsTest()*. Metoda spustí skript *hststest.py* a z výstupu zaznamená řetězec *"HSTS"* a nastaví proměnnou *isHsts* na **Pravdu**. Pokud ho nezaznamená, nastaví se **Nepravda**. Metodu lze vidět ve výpisu 2.19.

Výpis 2.19: Výpis metody runHstsTest().

```
private function runHstsTest(): void {
    $this->isHsts = FALSE;
    $command = escapeshellcmd('../bin/hststest.py '.$this->
        url);
    $out = shell_exec($command);
    $out_new = str_replace("\n", '', $out);
    if ($out_new == "HSTS") {
        $this->isHsts = TRUE;
        $this->score += 15;
    }
    else {
        $this->isHsts = FALSE;
        $this->set_vulnerability("HSTS");
    }
}
```

### Předání proměnných v presenteru ProtocolsCiphersPresenter

Proměnnou *isHsts* je dále potřeba předat do šablony v metodě *renderDefault()* v presenteru **ProtocolsCiphersPresenter**. Předání lze vidět ve výpisu 2.20.

Výpis 2.21: Výpis metody runCiphers().

```
private function runCiphers(): void {
    $command = escapeshellcmd('../bin/openssl.py␣'.$this->\
url.':443');
    $command_openssl = 'openssl␣version';
    #$out = shell_exec($command);
    #$out_new = str_replace("\n", '', $out);
    $this->openssl_version = shell_exec($command_openssl);
    $test = "ssl3:RC4-64-MD5,tls1:PSK-NULL-SHA,\
tls1:DHE-DSS-RC4-SHA,tls1:EXP1024-DES-CBC-SHA,\
tls1_1:ECDHE-RSA-AES256-SHA,tls1_1:RC4-MD5,\
tls1_2:ECDHE-RSA-AES256-GCM-SHA384,\
tls1_3:TLS_AES_128_GCM_SHA256,tls1_3:TLS_AES_128_CCM_SHA256";
    $this->fill_protocols_and_ciphers_to_arrays($test);
}
```

Výpis 2.20: Výpis předání proměnné isHsts do šablony.

```
$this->template->isHsts = $this->protocolsCiphersManager->
    isHsts;
```

### 2.6.7 Testování podporovaných kryptografických balíčků

Dalším postupem je testování kryptografických balíčků. Skript *openssl.py* byl zmíněn v kapitole 2.3.2, kde působil na začátku jako první skript pro funkčnost celé webové aplikace. Výstupem tedy je seznam podporovaných kryptografických balíčků na vzdáleném serveru. Výpis skriptu je přiložen v příloze B.1.

#### Metody v modelu ProtocolsCiphersManager

V modelu **ProtocolsCiphersManager** se vytvoří nová metoda *runCiphers()* pro spuštění skriptu *openssl.py*, do které se vloží obsah z metody *runTest()*. Ukázka metody *runCiphers()* je ve výpisu 2.21. Metoda *runTest()* bude později sloužit k jinému účelu.

#### Předání proměnných v presenteru ProtocolsCiphersPresenter

Všechny proměnné se předají v presenteru stejným způsobem jako ve výpisu 2.20, aby se s nimi mohlo pracovat v šabloně.

### 2.6.8 Spouštění metod v modelu ProtocolsCiphersManager

Metoda *runTest()* bude prezentovat hlavní spustitelnou metodu pro všechny testy. Ve výpisu 2.22 lze vidět posloupnost spouštění metod. První metoda je *runPing()*, která vrací **Pravdu**, jestliže je vzdálený server dostupný. Pokud podmínka platí, spouští se metoda *runInfoScan()*, která po detekci HTTPS také vrátí **Pravdu**. Nakonec se při platné podmínce spustí metody *runCertInfo()*, *runHstsTest()* a *runCiphers()*.

Výpis 2.22: Výpis hlavní spustitelné metody runTest().

```
// Hlavní spustitelná metoda
public function runTest($url): void {
    // Naplnění zranitelností z JSON do paměti
    $json = file_get_contents("../bin/vuln_info.json");
    $this->vulnMapping = Json::decode($json,
        Json::FORCE_ARRAY);
    // Datum a čas testování
    $this->timeNow = date("Y-m-d␣H:i:s");
    // Start testování
    $this->setUrl($url);
    if ($this->runPing()) {
        if ($this->runInfoScan()) {
            $this->runCertInfo();
            $this->runHstsTest();
            $this->runCiphers();
        }
    }
}
```

### 2.6.9 Detekce zranitelností

Detekce zranitelností je ve webové aplikaci řešena zároveň při volání metod, které naplňují proměnné z výstupu daného skriptu. Například ve výpisu F.3 metoda *get_info_about_server()* testuje i stavy portů, a pokud je port otevřený a z hlediska bezpečnosti by otevřený být neměl, tak se metodou *set_vulnerability()* nastaví zranitelnost. Metodu *set_vulnerability()* lze vidět ve výpisu 2.23 a jejím argumentem je pojmenovaná zranitelnost. Řetězec neboli pojmenovaná zranitelnost se v metodě vloží do pole tak, aby nevznikaly duplicity. Toto pole představuje seznam všech detekovaných zranitelností a plní se zároveň během postupu testování.

Metoda *fill_protocols_and_ciphers_to_arrays()*, která byla zmíněna v kapitole 2.4, se upraví pomocí podmínek podle výpisu v příloze F.5, aby splňovala funkci

pro detekci zranitelností. Přidá se také metoda *set_vulnerability()* u všech zjištěných informací.

Podobně se metoda *set_vulnerability()* implementuje u všech detekovaných informací podle tabulky 2.2. Také se podle této tabulky vytvoří soubor *vuln_info.json* a v něm se vytvoří struktura podle výpisu 2.24. Výpis neobsahuje seznam všech zranitelností v tabulce 2.2, ale jen část pro ukázku. Ze souboru ve formátu *.json* lze jednoduše číst hodnoty na základě klíče. Například klíč *"DROWN Útok"* obsahuje pole s dalšími klíči *description*, *html* a *htmlText* s vlastními hodnotami. Datový formát JSON (JavaScript Object Notation) slouží pro jednoduchou výměnu dat mezi nezávislými a naprosto odlišnými systémy nebo aplikacemi [68].

Klíč v souboru *vuln_info.json* představuje pojmenovanou zranitelnost. U každé zranitelnosti klíč *description* obsahuje popis a vysvětlení, proč se jedná o zranitelnost. Může obsahovat také popsané doporučení. Klíč *html* obsahuje externí odkaz, kde lze vyhledat podrobnější informace o zranitelnosti nebo odkaz na dané doporučení. Klíč *htmlText* pouze obsahuje pojmenování odkazu v šabloně při zobrazení.

Obsah uložený v souboru *vuln_info.json* je potřeba načíst do webové aplikace. To se provede v hlavní spustitelné metodě *runTest()* pomocí nástroje **Nette\Utils\Json** metodou *Json::decode()* [69] a uloží se do proměnné *vulnMapping*. Načtení souboru a použití nástroje **Json** je znázorněno ve výpisu 2.22.

Výpis 2.23: Výpis metody set_vulnerability().

```
private function set_vulnerability(string $vuln): void {
    if (!in_array($vuln, $this->vulnerabilities)) {
        array_push($this->vulnerabilities, $vuln);
    }
}
```

Výpis 2.24: Částečný výpis souboru vuln_info.json.

```
{
    ...
    "Slabý klíč": {
        "description":"Byl použit slabý veřejný klíč.",
        "html":"https://www.keylength.com/en/4/",
        "htmlText":"Doporučení pro délky klíčů (NIST)"
    },
    "NULL": {
        "description":"TLS spojení není šifrované.",
        "html":"https://tools.ietf.org/html/rfc4785",
        "htmlText":"RFC 4785"
    },
```

```
    "DROWN␣Útok": {
        "description":"Byl␣použitý␣protokol␣SSL␣2.0.",
        "html":"https://drownattack.com",
        "htmlText":"Informace"
    },
    ...
}
```

## 2.7 Hodnocení bezpečnosti testovaného serveru

V této kapitole bude vysvětlen postup a rozdělení výsledků od testovaného vzdáleného serveru do kategorií. Bude také popsán způsob implementace udělování bodového skóre serveru.

### 2.7.1 Kategorie a rozdělení výsledku z testování

Na základě všech zjištěných zranitelností je testovaný vzdálený server hodnocen od 0 do 100 bodů. Podle tabulky 2.3 lze vidět rozdělení od kategorie **Dobře zabezpečený** do kategorie **Zranitelný** s bodovými intervaly a barevnou reprezentací. Do kategorie **Zranitelný** může spadat server, který má velice slabé zabezpečení nebo žádné.

### 2.7.2 Přiřazení bodového skóre ke zjištěným informacím

Zjištěné informace se obodují podle tabulky 2.4. Maximální dosažitelná hodnota skóre je 100 bodů. U minimální hodnoty záleží, kolik zranitelností se detekuje. Avšak kdyby bodové skóre vyšlo v záporných hodnotách, tak se zobrazené skóre přepočítá na 0.

Bodový systém se implementuje do webové aplikace proměnnou *score*, do které se vždy při zjištění informace podle tabulky 2.4 přičte nebo odečte hodnota. Ukázka implementace je znázorněna například ve výpisu 2.19 u mechanismu HSTS nebo ve výpisu v příloze F.4 v metodě *run_cert_info()*.

## 2.8 Vizuální implementace výsledků vyhodnoceného serveru

V této kapitole bude navrženo rozložení výsledků vyhodnoceného serveru na webové stránce **Appserveru**. Dále bude návrh implementován do webové šablony.

Tab. 2.2: Seznam pojmenovaných zranitelností.

| **Pojmenovaná zranitelnost** | **Popis zranitelností** |
|---|---|
| LUCKY13 | Byl použit mód CBC při šifrování. |
| HSTS | Nedetekováno - potenciální útok MITM. |
| HTTPS | Server není zabezpečen SSL/TLS spojením. |
| HTTP | Server má povolený nezabezpečený port 80. Hrozí potenciální odposlech spojení |
| 21 | Server má otevřený port 21 (FTP). Potenciální útok uhádnutím hesla. |
| 22 | Server má otevřený port 22 (SSH). Potenciální útok uhádnutím hesla. |
| 23 | Server má otevřený port 23 (Telnet). Potenciální útok uhádnutím hesla. |
| 113 | Server má otevřený port 113. Potenciální útok odposlechu identifikačních a autentizačních údajů. |
| 3389 | Server má otevřený port 3389 (RDP). Otevřený přístup ke vzdálené ploše. |
| Renegotiation | Server nepodporuje bezpečné opětovné vyjednávání. |
| Slabý klíč | Byl použit slabý veřejný klíč. |
| SHA1 | Byl použit slabý hash SHA1. |
| Neplatný certifikát | Platnost certifikátu vypršela. |
| RC | Byl použit slabý šifrovací algoritmus RC2 nebo RC4. |
| MD5 | Byl použit slabý hash MD5. |
| Výměna klíče přes RSA | Výměna klíče proběhla pomocí RSA algoritmu. |
| NULL | TLS spojení není šifrované. |
| DES | Bylo použité slabé šifrování DES nebo 3DES. |
| DSS | Byla použitá slabá autentizace DSS. |
| BEAST Útok | Byly použitý protokoly TLS 1.0 a starší. |
| POODLE Útok | Byl použitý protokol SSL 3.0. |
| RACCOON Útok | Výměna klíče proběhla pomocí DH nebo DHE algoritmu. |
| DROWN Útok | Byl použitý protokol SSL 2.0. |
| Logjam Útok | Výměna klíče proběhla se slabým šifrováním DHE_EXPORT. |
| CRIME | Server používá kompresi. |

Tab. 2.3: Kategorie hodnocení serveru.

| **Kategorie** | **Známka** | **Body** | **Barevná reprezentace** |
|---|---|---|---|
| Dobře zabezpečený | A+ | 100 - 95 | Zelená |
| Zabezpečený | A | 94 - 80 | Zelená |
| Částečně zabezpečený | B | 79 - 50 | Oranžová |
| Zranitelný | C | 49 - 0 | Červená |

Tab. 2.4: Bodové přiřazení ke zjištěným informacím.

| Zjištěná informace | | Bodové skóre |
|---|---|---|
| Porty | 113 | -5 |
| | 21 | -5 |
| | 22 | -10 |
| | 23 | -10 |
| | 3389 | -10 |
| | 80 | -3 |
| | HTTPS | 50 |
| Podpora SSL/TLS | SSL 2 | -15 |
| | SSL 3 | -15 |
| | TLS 1.0 | -10 |
| | TLS 1.2 | 10 |
| | TLS 1.3 | 20 |
| HSTS | | 15 |
| Secure Renegotiation | | 5 |
| Slabý veřejný klíč | | -10 |
| Slabý podpis | | -5 |
| Neplatný certifikát | | -10 |
| Zapnutá komprese (CRIME) | | -5 |

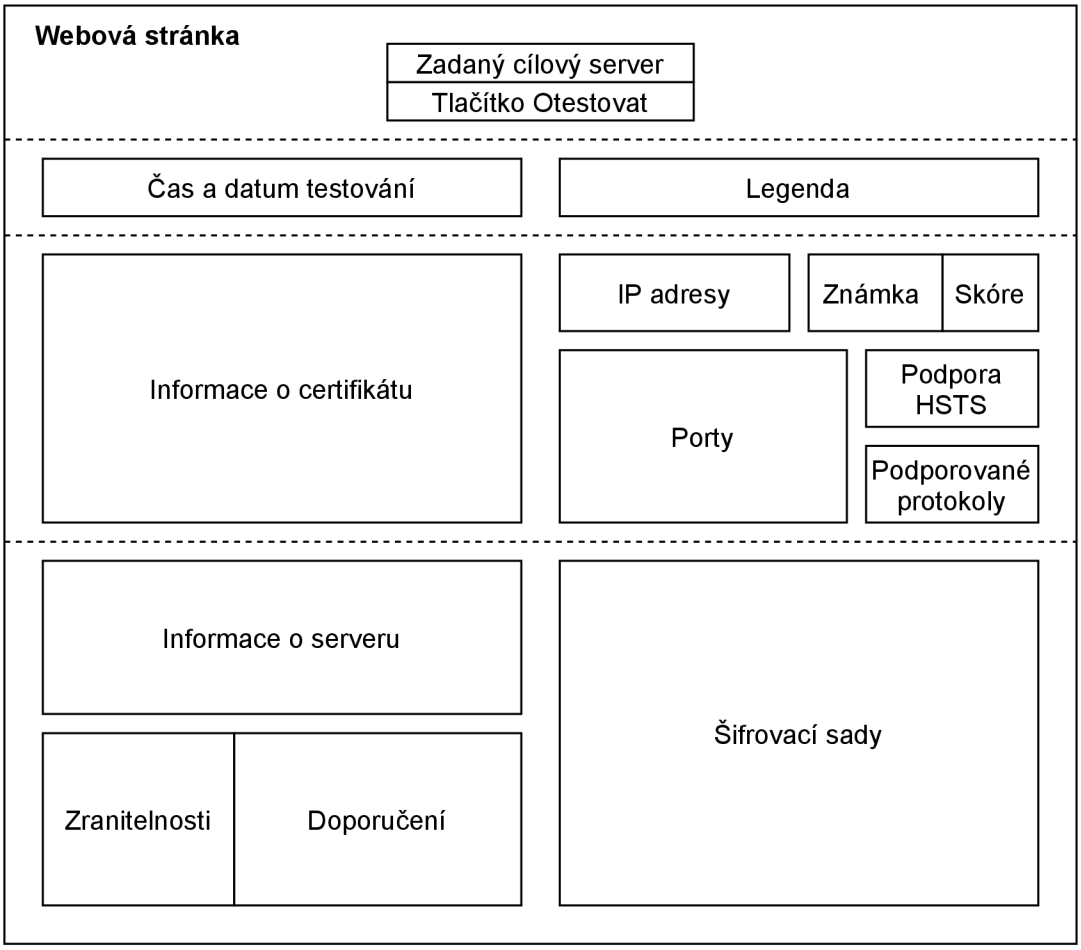
### 2.8.1 Návrh rozložení výsledků na webové stránce

Rozložení webové stránky pro zobrazení výsledků testovaného vzdáleného serveru je vidět na obrázku 2.5. Rozložení bylo navrženo do čtyř částí rozdělených čárkovanou čárou. První horní část je pro zadání vzdáleného serveru a tlačítko pro spuštění testu. Při prvním načtení stránky je vidět pouze tato první část. Po kliknutí na tlačítko se na základě testu zobrazí další části, ale jenom v případě, že byl server zadaný ve správném tvaru a je dostupný.

Druhá část je pouze informativního charakteru. Na pravé straně je oblast, která vypíše název zadaného vzdáleného serveru a čas a datum testování. Na levé straně je legenda, která vysvětluje dosažené bodové skóre.

Třetí oddělená část zobrazuje na pravé straně informace o certifikátu. Na levé straně zobrazí všechny IP adresy spojené s doménou, zobrazí stavy testovaných portů, známku se skórem, zda podporuje HSTS a seznam podporovaných SSL/TLS protokolů.

Ve čtvrté spodní části zobrazí informace o serveru a zranitelnosti s doporučením, které jsou na pravé straně. Na levé straně zobrazí seznam všech podporovaných kryptografických balíčků.




Obr. 2.5: Návrh rozložení výsledků testovaného serveru.

### 2.8.2 Implementace návrhu do šablony webové aplikace

Jednotlivé části popsané v podkapitole 2.8.1 reprezentují řádky a pravou a levou stranu představují sloupce v šabloně *default.latte*. Struktura šablony je ukázána

v příloze G.1. Rozvržené oblasti jsou HTML tabulky dosazené za komentáře ve vytvořené struktuře.

Komentář *Čas a datum testování* se nahradí výpisem G.2. Komentář *Legenda* je ukázána ve výpisu G.3. V další části je *Informace o certifikátu*, kde je tabulka ukázána ve výpisu G.4, oblast *IP adresy* ukázány ve výpisu G.5, *Známka a skóre* jsou zobrazeny ve výpisu G.6 a oblast *Porty*, které lze vidět ve výpisu G.7. Dále je pak zobrazena dvojice tabulek *Podpora HSTS* ve výpisu G.8 a *Podporované protokoly* ve výpisu G.9. V poslední části je dvojice tabulek *Informace o serveru* zobrazené ve výpisu G.10 a *Zranitelnosti a doporučení* ve výpisu G.11. Nakonec tabulku oblasti *Šifrovací sady* lze vidět ve výpisu G.12. Všechny použité proměnné v šabloně *default.latte* musejí být vypsané v presenteru **ProtocolsCiphersPresenter** v metodě *renderDefault()*.

Výsledná implementace rozložení výsledků na webové stránce je vidět na obrázku v příloze H.1.

## 2.9 Analýza testovaných vzdálených serverů

Vytvořenou webovou aplikací Appserver bylo otestováno několik vzdálených serverů a jejich hodnocení bylo porovnáno s webovou aplikací SSL Labs. Taktéž se pomocí ladícího nástroje Tracy měřila doba testování Appserveru. V tabulce 2.5 lze vidět seznam serverů s dobou trvání testu, která je u některých serverů dlouhá. Například test serveru *czc.cz* trval přibližně 183 vteřin, tedy trojnásobně déle než u serveru *seznam.cz*, který trval pouze 63 vteřin.

Celková doba testu může záviset buď na rychlosti zpracování požadavků vzdáleného serveru při testování kryptografických balíčků, nebo geolokaci fyzického serveru. Dobu trvání testu může také ovlivnit umístění webové aplikace Appserver. Pokud by aplikace běžela na dostupnějším místě než ve virtuálním prostředí v lokální síti, tak by byla odezva komunikace určitě menší.

Při analýze bylo dále monitorováno využití procesoru během testování vzdáleného serveru, které se pohybovalo kolem 10 % s dvěma jádry a přibližně 25 % s jedním jádrem procesoru AMD Ryzen 5 1400. Využití operační paměti nebylo při testování nijak viditelně ovlivněno.

Tab. 2.5: Tabulka testovaných vzdálených serverů.

| **Vzdálený server** | **Doba testu Appserveru [s]** | **Známka od Appserveru** | **Známka od SSL Labs** |
|---|---|---|---|
| webserver-actinver-prd.lfr.cloud | 47,74 | A | A+ |
| seznam.cz | 63,04 | B | A |
| vutbr.cz | 56,71 | C | B |
| fekt.vut.cz | 45,93 | C | B |
| zebrabrno.cz | 52,60 | C | T (nedůvěryhodný) |
| google.cz | 49,30 | B | B |
| k-lifepartners.co.jp | 337,96 | C | T (nedůvěryhodný) |
| czc.cz | 183,22 | B | A+ |
| developer.huawei.com | 65,45 | A | A+ |

# Závěr

V teoretické části bylo na začátku popsáno několik šifrovacích algoritmů a hašovacích funkcí. Dále jsou popsány jednotlivé verze protokolů SSL/TLS pro zabezpečené spojení přes HTTP včetně průběhu navazování spojení a jejich zranitelnosti, které využívají útoky popsané v podkapitole 1.4 s doporučenou obranou. V kapitole 1.5 jsou popsaná současná doporučení a standardy na TLS spojení a použití kryptografických algoritmů s délkami klíčů, které by měli vývojáři a jiné osoby zodpovědné za bezpečnost dodržovat. V následující kapitole 1.6 jsou uvedena současná řešení pro testování bezpečnosti webového serveru, dále popsané nástroje OpenSSL, Nmap a Curl pro testování vzdálených serverů. Také jsou popsané použité technologie při vývoji aplikace v praktické části této diplomové práce.

Na konci teoretické části je vysvětlen model MVC jako vzor pro vývoj webových aplikací a použitý framework Nette, který z tohoto modelu vychází.

V praktické části bylo připraveno prostředí pro vývoj a instalace PHP. Dále se pomocí Composeru připravila základní struktura webové aplikace frameworku Nette. Poté byl vytvořen návrh webové aplikace s postupem testování a její implementace s prvním zkušebním skriptem. Po úspěšné implementaci prvního skriptu byly implementovány další skripty podle postupu testování, který je vidět na obrázku 2.2. Skripty byly napsány v jazyce Python. Zjištěné informace se zobrazí na webové stránce, která je zobrazena na obrázku v příloze H.1 a ukazuje grafické rozložení výsledků testovaného serveru včetně zobrazení zranitelností s doporučeními. Vzdálený server je nakonec klasifikován do kategorie bezpečnosti od A+ až C.

Webová aplikace s implementovanými skripty ve frameworku Nette je plně funkční. Zobrazuje grafické znázornění bezpečnosti testovaného serveru včetně detekcí minimálně pěti různých zranitelností a lze tedy konstatovat, že všechny stanovené cíle diplomové práce byly splněny.

Jako možné rozšíření webové aplikace je implementace databáze, do které by se uchovávala historie testovaných serverů a umožnit tak jejich prohlížení. Také lze aplikaci rozšířit o možnost exportovat výsledky do formátů JSON, XML (Extensible Markup Language) pro výměnu dat či PDF (Portable Document Format) pro prezentaci výsledků.

# Literatura


[1] BURDA, K. *Kryptografie okolo nás*. Edice CZ.NIC, Praha, 2019, [cit. 25. 11. 2020]. ISBN 978-80-88168-52-2. Dostupné z URL: <https://knihy.nic.cz/files/edice/Kryptografie_okolo_nas.pdf>.

[2] NIST. *Digital Signature Standard (DSS)* [online]. National Institute of Standards and Technology, FIPS 186-4, 2013, [cit. 15. 05. 2021]. Dostupné z URL: <https://csrc.nist.gov/publications/detail/fips/186/4/final>.

[3] RIVEST, R.L., SHAMIR, A., and ADLEMAN, L. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*. Commun. ACM 21, 1978, [cit. 25. 11. 2020]. ISBN 978-80-88168-52-2. Dostupné z URL: <https://doi.org/10.1145/359340.359342>.

[4] RESCORLA, E., *Diffie-Hellman Key Agreement Method* [online]. RFC 2631, DOI 10.17487/RFC2631, 1999, [cit. 25. 11. 2020]. Dostupné z URL: <https://www.rfc-editor.org/info/rfc2631>.

[5] BRUGGER, Max F. and FREDERICK, Christina A. *The Discrete Logarithm Problem and Ternary Functional Graphs* [online]. Mathematical Sciences Technical Reports (MSTR). 2007, [cit. 25. 11. 2020]. Dostupné z URL: <https://scholar.rose-hulman.edu/math_mstr/47>.

[6] Standards for Efficient Cryptography, *SEC 1: Elliptic Curve Cryptography* [online]. Version 2.0, 2009, [cit. 25. 11. 2020]. Dostupné z URL: <https://www.secg.org/sec1-v2.pdf>.

[7] Crypto-IT, *RC4 stream cipher with symmetric secret key* [online]. 2020, [cit. 25. 11. 2020]. Dostupné z URL: <http://www.crypto-it.net/eng/symmetric/rc4.html>.

[8] DELFS, H., KNEBL, H. *Introduction to Cryptography: Principles and Applications.*. Third Edition, Springer-Verlag Berlin Heidelberg, 2015, [cit. 25. 11. 2020]. ISBN 978-3-662-479 4-2.

[9] LAKE, J. *What is 3DES encryption and how does DES work?* [online]. Comparitech, 2019, [cit. 25. 11. 2020]. Dostupné z URL: <https://www.comparitech.com/blog/information-security/3des-encryption/>.

[10] CryptoSys, *Triple DES cryptography software* [online]. 2020, [cit. 26. 11. 2020]. Dostupné z URL: <https://www.cryptosys.net/3des.html>.

[11] DWORJIN, J., M. *Recommendation for Block Cipher Modes of Operation: Methods and Techniques* [online]. NIST, SP 800-38A, 2001, [cit. 26. 11. 2020]. Dostupné z URL:
<https://csrc.nist.gov/publications/detail/sp/800-38a/final>.

[12] DWORJIN, J., M. *Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality* [online]. NIST, SP 800-38C, 2007, [cit. 26. 11. 2020]. Dostupné z URL:
<https://csrc.nist.gov/publications/detail/sp/800-38a/final>.

[13] DWORJIN, J., M. *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC* [online]. NIST, SP 800-38D, 2007, [cit. 26. 11. 2020]. Dostupné z URL:
<https://csrc.nist.gov/publications/detail/sp/800-38d/final>.

[14] Tým podpory SSL, *Co je kryptografická funkce hash?* [online]. SSL.com, 2015, [cit. 26. 11. 2020]. Dostupné z URL:
<https://www.ssl.com/cs/Nejčastější-dotazy/co-je-kryptografická-hashovací-funkce/>.

[15] RISTIC, Ivan. *Bulletproof SSL and TLS: Understanding and Deploying SSL/TLS and PKI to Secure Servers and Web Applications.* Feisty Duck, 2015, [cit. 20. 11. 2020]. ISBN: 978-1907117046.

[16] RIVEST, R., *The MD5 Message-Digest Algorithm* [online]. RFC 1321, DOI 10.17487/RFC1321, 1992, [cit. 26. 11. 2020]. Dostupné z URL:
<https://www.rfc-editor.org/info/rfc1321>.

[17] H. KUMAR et al., *Rainbow table to crack password using MD5 hashing algorithm.* 2013 IEEE Conference on Information & Communication Technologies, Thuckalay, Tamil Nadu, India, 2013, doi: 10.1109/CICT.2013.6558135, [cit. 26. 11. 2020].

[18] CWI Amsterdam, Google Research, *The first collision for full SHA-1* [online]. 2020, [cit. 21. 11. 2020]. Dostupné z URL:
<https://shattered.io>.

[19] NIST. *Secure Hash Standard (SHS)* [online]. National Institute of Standards and Technology FIPS 180-4, 2015, [cit. 26. 11. 2020]. Dostupné z URL:
<https://csrc.nist.gov/publications/detail/fips/180/4/final>.

[20] NIST. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions* [online]. National Institute of Standards and Technology FIPS 202,

2015, [cit. 20. 05. 2021]. Dostupné z URL:
<https://csrc.nist.gov/publications/detail/fips/202/final>.

[21] Web security blog, *Historie SSL/TLS certifikátů* [online]. 2018, [cit. 20. 11. 2020]. Dostupné z URL:
<https://blog.sslmentor.cz/clanky/historie-ssl-tls-certifikatu/>.

[22] TURNER, S. and POLK, T. *Prohibiting Secure Sockets Layer (SSL) Version 2.0* [online]. RFC 6176, 2011, [cit. 20. 11. 2020]. Dostupné z URL:
<https://www.rfc-editor.org/info/rfc6176>.

[23] BARNES, R., THOMSON, M., PIRONTI, A., and LANGLEY A. *Deprecating Secure Sockets Layer Version 3.0* [online]. RFC 7568, 2015, [cit. 20. 11. 2020]. Dostupné z URL:
<https://www.rfc-editor.org/info/rfc7568>.

[24] MOELLER, B. *This POODLE bites: exploiting the SSL 3.0 fallback* [online]. 2014, [cit. 20. 11. 2020]. Dostupné z URL:
<http://googleonlinesecurity.blogspot.com/2014/10/this-poodle-bites-exploiting-ssl-30.html>.

[25] MORIARTY, K., FARRELL, S. *Deprecating TLSv1.0 and TLSv1.1* [online]. 2020, [cit. 21. 11. 2020]. Dostupné z URL:
<https://tools.ietf.org/html/draft-ietf-tls-oldversions-deprecate-09>.

[26] DIERKS, T. and RESCORLA E. *The Transport Layer Security (TLS) Protocol Version 1.2* [online]. RFC 5246, 2008, [cit. 21. 11. 2020]. Dostupné z URL:
<https://www.rfc-editor.org/info/rfc5246>.

[27] RESCORLA, E. *The Transport Layer Security (TLS) Protocol Version 1.3* [online]. RFC 8446, DOI 10.17487/RFC8446, 2018, [cit. 22. 11. 2020]. Dostupné z URL:
<https://www.rfc-editor.org/info/rfc8446>.

[28] FIELDING, R., GETTYS, J., MOGUL, J., FRYSTYK, H., MASINTER, L., LEACH, P., and BERNERS-LEE, T., *Hypertext Transfer Protocol – HTTP/1.1* [online]. RFC 2616, DOI 10.17487/RFC2616, 1999, [cit. 27. 11. 2020]. Dostupné z URL:
<https://www.rfc-editor.org/info/rfc2616>.

[29] RESCORLA, E. *HTTP Over TLS* [online]. RFC 2818, DOI 10.17487/RFC2818, 2000, [cit. 27. 11. 2020]. Dostupné z URL: <https://www.rfc-editor.org/info/rfc2818>.

[30] NIST. *CVE-2012-4929 Detail* [online]. National Institute of Standards and Technology, National Vulnerability Database, CVE-2012-4929, 2012, [cit. 11. 04. 2021]. Dostupné z URL: <https://nvd.nist.gov/vuln/detail/CVE-2012-4929>.

[31] Crashtest Security GmbH. *Prevent SSL CRIME* [online]. Vulnerabilities requiring reconfiguration, 2021, [cit. 11. 04. 2021]. Dostupné z URL: <https://wiki.crashtest-security.com/prevent-ssl-crime>.

[32] NIST. *CVE-2013-0169 Detail* [online]. National Institute of Standards and Technology, National Vulnerability Database, CVE-2013-0169, 2013, [cit. 13. 04. 2021]. Dostupné z URL: <https://nvd.nist.gov/vuln/detail/CVE-2013-0169>.

[33] c0D3M. *Lucky 13 Attack Explained* [online]. A Medium Corporation, 2019, [cit. 13. 04. 2021]. Dostupné z URL: <https://medium.com/@c0D3M/lucky-13-attack-explained-dd9a9fd42fa6>.

[34] Crashtest Security GmbH. *Prevent SSL LUCKY13* [online]. Vulnerabilities requiring reconfiguration, 2021, [cit. 11. 04. 2021]. Dostupné z URL: <https://wiki.crashtest-security.com/prevent-ssl-lucky13>.

[35] Synopsys, Inc. *The Heartbleed Bug* [online]. 2020, [cit. 28. 11. 2020]. Dostupné z URL: <https://heartbleed.com>.

[36] Adrian D., Bhargavan K., Durumeric Z., Gaudry P., Green M., Halderman A. J., Heninger N., Springall D., Thomé E., Valenta L., VanderSloot B., Wustrow E., Zanella-Béguelin S., Zimmermann P. *Weak Diffie-Hellman and the Logjam Attack* [online]. Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice. Denver, 2015, [cit. 17. 04. 2021]. Dostupné z URL: <https://weakdh.org/>.

[37] Crashtest Security GmbH. *Prevent SSL LOGJAM* [online]. Vulnerabilities requiring reconfiguration, 2021, [cit. 17. 04. 2021]. Dostupné z URL: <https://wiki.crashtest-security.com/prevent-ssl-logjam>.

[38] Aviram N., Schinzel S., Somorovsky J., Heninger N., Dankel M., Steube J., Valenta L., Adrian D., Halderman A. J., Dukhovni V., Käsper E., Cohney S., Engels S., Paar Ch. and Shavitt Y. *The DROWN Attack* [online]. DROWN: Breaking TLS using SSLv2. 2016, [cit. 17. 04. 2021]. Dostupné z URL: <https://drownattack.com/>.

[39] Merget R., Brinkmann M., Aviram N., Somorovsky J., Mittmann J. and Schwenk J. *Raccoon Attack* [online]. Raccoon Attack: Finding and Exploiting Most-Significant-Bit-Oracles in TLS-DH(E). 2021, [cit. 18. 04. 2021]. Dostupné z URL: <https://raccoon-attack.com/>.

[40] The SSL Store™. *Raccoon Attack: Researchers Find A Vulnerability in TLS 1.2* [online]. Hashedout by The SSL Store™ Blog, 2020, [cit. 18. 04. 2021]. Dostupné z URL: <https://www.thesslstore.com/blog/raccoon-attack-researchers-find-a-vulnerability-in-tls-1-2/>.

[41] EXPLOIT DATABASE, *Hijacking Web 2.0 Sites with SSLstrip—Hands-on Training* [online]. 2010, [cit. 22. 11. 2020]. Dostupné z URL: <https://www.exploit-db.com/exploits/15377>.

[42] HODGES, J., JACKSON, C., and BARTH, A. *HTTP Strict Transport Security (HSTS)* [online]. RFC 6797, DOI 10.17487/RFC6797, 2012, [cit. 22. 11. 2020]. Dostupné z URL: <https://www.rfc-editor.org/info/rfc6797>.

[43] NIST. *Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations* [online]. National Institute of Standards and Technology, SP 800-52 Rev. 2, 2019, [cit. 15. 05. 2021]. Dostupné z URL: <https://csrc.nist.gov/publications/detail/sp/800-52/rev-2/final>.

[44] NIST. *Transitioning the Use of Cryptographic Algorithms and Key Lengths* [online]. National Institute of Standards and Technology, SP 800-131A Rev. 2, 2019, [cit. 15. 05. 2021]. Dostupné z URL: <https://csrc.nist.gov/publications/detail/sp/800-131a/rev-2/final>.

[45] NIST. *Recommendation for Key Management: Part 1 – General* [online]. National Institute of Standards and Technology, SP 800-57 Part 1 Rev. 5, 2020, [cit. 15. 05. 2021]. Dostupné z URL:

<https://csrc.nist.gov/publications/detail/sp/800-57-part-1/rev-5/final>.

[46] NÚKIB. *MINIMÁLNÍ POŽADAVKY NA KRYPTOGRAFICKÉ ALGORITMY* [online]. Národní úřad pro kybernetickou a informační bezpečnost, Doporučení v oblasti kryptografických prostředků. 2018, [cit. 15. 05. 2021]. Dostupné z URL:
<https://www.nukib.cz/cs/infoservis/doporuceni/1504-doporuceni-v-oblasti-kryptografickych-prostredku/>.

[47] ENISA. *Algorithms, key size and parameters report – 2014* [online]. European Union Agency for Cybersecurity. 2014, [cit. 15. 05. 2021]. Dostupné z URL:
<https://www.enisa.europa.eu/publications/algorithms-key-size-and-parameters-report-2014>.

[48] Qualys, Inc. *SSL Labs* [online], HOW WELL DO YOU KNOW SSL? 2021, [cit. 3. 5. 2021]. Dostupné z URL:
<https://www.ssllabs.com>.

[49] drwetter. *Testing TLS/SSL encryption* [online], 2021, [cit. 4. 5. 2021]. Dostupné z URL:
<https://testssl.sh>.

[50] OWASP® Foundation. *OWASP, Open Web Application Security Project* [online], Projects: OWASP Top Ten and OWASP Web Security Testing Guide. 2021, [cit. 9. 5. 2021]. Dostupné z URL:
<https://owasp.org>.

[51] OpenSSL Software Foundation, *Cryptography and SSL/TLS Toolkit* [online]. 2018, [cit. 29. 11. 2020]. Dostupné z URL:
<https://www.openssl.org>.

[52] NMAP.ORG, *Nmap "Network Mapper"* [online]. 2020, [cit. 1. 12. 2020]. Dostupné z URL:
<https://nmap.org>.

[53] curl, *curl://* [online]. 2021, [cit. 20. 02. 2021]. Dostupné z URL:
<https://curl.se>.

[54] Python Software Foundation, *python™* [online]. 2020, [cit. 1. 12. 2020]. Dostupné z URL:
<https://www.python.org>.

[55] Microsoft, *https://code.visualstudio.com* [online]. 2020, [cit. 1. 12. 2020]. Dostupné z URL:
<https://code.visualstudio.com>.

[56] The PHP Group, *PHP Manual* [online]. 2020, [cit. 2. 12. 2020]. Dostupné z URL:
<https://www.php.net/manual/en/index.php>.

[57] NEMEČEK, A., *Najpoužívanejšie webové servery – Apache vs. Nginx.* [online]. WebSupport blog, 2019, [cit. 1. 12. 2020]. Dostupné z URL:
<https://www.websupport.sk/blog/2019/01/najpouzivanejsie-webove-servery-porovnanie-apache-vs-nginx/>.

[58] The Apache Software Foundation, *Apache HTTP Server* [online]. 2020, [cit. 2. 12. 2020]. Dostupné z URL:
<https://httpd.apache.org>.

[59] nginx, *nginx [engine x]* [online]. 2020, [cit. 2. 12. 2020]. Dostupné z URL:
<http://nginx.org/en/>.

[60] Tutorialspoint, *MVC Framework - Introduction* [online]. 2020, [cit. 2. 12. 2020]. Dostupné z URL:
<https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm>.

[61] Nette Foundation, *Nette* [online]. 2020, [cit. 2. 12. 2020]. Dostupné z URL:
<https://nette.org/cs/>.

[62] ComputingforGeeks, *How To Install PHP 7.4 on Debian 10 / Debian 9* [online]. 2020, [cit. 11. 10. 2020]. Dostupné z URL:
<https://computingforgeeks.com/how-to-install-latest-php-on-debian/>.

[63] A Dependency Manager for PHP, *Download Composer, Command-line installation* [online]. 2020, [cit. 11. 10. 2020]. Dostupné z URL:
<https://getcomposer.org/download/>.

[64] Nette Foundation, *Dokumentace Nette 3.0, Řešení častých dotazů* [online]. 2020, [cit. 11. 10. 2020]. Dostupné z URL:
<https://doc.nette.org/cs/3.0/troubleshooting#toc-nastaveni-prav-adresaru>.

[65] Nette Foundation, *Dokumentace Nette 3.0, Píšeme první aplikaci!* [online]. 2020, [cit. 31. 10. 2020]. Dostupné z URL:
<https://doc.nette.org/cs/3.0/quickstart/getting-started>.

[66] CASTRO, Elizabeth a Bruce HYSLOP. *HTML5 a CSS3: názorný průvodce tvorbou WWW stránek.* Brno: Computer Press, 2012, [cit. 12. 11. 2020]. ISBN 978-80-251-3733-8.

[67] Bootstrap team, *Documentation, Overview* [online]. 2020, [cit. 12. 11. 2020]. Dostupné z URL: <https://getbootstrap.com/docs/4.5/layout/overview/>.

[68] JSON, *Introducing JSON* [online]. 1999, [cit. 17. 05. 2021]. Dostupné z URL: <https://www.json.org/json-en.html>.

[69] Nette Foundation, *Dokumentace Nette 3.0, Práce s JSON* [online]. 2021, [cit. 2. 5. 2021]. Dostupné z URL: <https://doc.nette.org/cs/3.0/json>.

# Seznam symbolů, veličin a zkratek

| | |
|---|---|
| **DSA** | Digital Signature Algorithm |
| **NIST** | Národní institut standardů a technologie |
| **ECDSA** | Elliptic Curve Digital Signature Algorithm |
| **RSA** | Rivest, Shamir, Adleman algoritmus |
| **DH** | Diffie-Hellman algoritmus |
| **DHE** | Diffie-Hellman Ephemeral algoritmus |
| **DES** | Data Encryption Standard |
| **EDE** | Encrypt-Decrypt-Encrypt |
| **AES** | Advanced Encryption Standard |
| **CBC** | Cipher Block Chaining |
| **IV** | Inicializační vektor |
| **CTR** | Counter |
| **CCM** | Counter with Cipher Block Chaining-Message Authentication Code |
| **GCM** | Galois/Counter Mode |
| **MD5** | Message-Digest algorithm |
| **SHA** | Secure Hash Algorithm |
| **SSL** | Secure Sockets Layer |
| **TLS** | Transport Layer Security |
| **TCP/IP** | Transmission Control Protocol/Internet Protocol |
| **OSI** | Open Systems Interconnection |
| **MITM** | Man-in-the-middle |
| **CBC** | Cipher block chaining |
| **RC4** | Rivest Cipher 4 - symetrická kryptografická šifra |
| **POODLE** | Padding Oracle On Downgraded Legacy Encryption |

| | |
|---|---|
| **MAC** | Message authentication code |
| **HTTP** | Hypertext Transfer Protocol |
| **HTTPS** | Hypertext Transfer Protocol Secure |
| **CRIME** | Compression Ratio Info-leak Made Easy |
| **DROWN** | Decrypting RSA with Obsolete and Weakened encryption |
| **BEAST** | Browser Exploit Against SSL/TLS |
| **DTLS** | Datagram Transport Layer Security |
| **UDP** | User Datagram Protocol |
| **HSTS** | HTTP Strict Transport Security |
| **ENISA** | Evropská agentura pro bezpečnost sítí a informací |
| **NÚKIB** | Národní úřad pro kybernetickou a informační bezpečnost |
| **OWASP** | Open Web Application Security Project |
| **URL** | Uniform Resource Locator |
| **Nmap** | Network Mapper |
| **VS Code** | Visual Studio Code |
| **MVC** | Model View Controller |
| **PHP** | PHP: Hypertextový preprocesor |
| **HTML** | Hypertext Markup Language |
| **CSS** | Cascading Style Sheets |
| **IP** | Internet Protocol |
| **ICMP** | Internet Control Message Protocol |
| **FTP** | File Transfer Protocol |
| **SSH** | Secure Shell |
| **DNS** | Domain Name System |
| **JSON** | JavaScript Object Notation |

| | |
|---|---|
| **XML** | Extensible Markup Language |
| **PDF** | Portable Document Format |

# Seznam příloh

# A Výpis hlavní šablony aplikace

Výpis A.1: Hlavní šablona aplikace.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content=
  "width=device-width, initial-scale=1, shrink-to-fit=no">
  <!-- Bootstrap CSS -->
  {include styles}
  <title>{ifset title}{include title|stripHtml} | {/ifset}
  Nette</title>
</head>

<body>

  <div n:foreach="$flashes as $flash" n:class=
  "flash, $flash->type">{$flash->message}</div>

  {include content}

  {block scripts}
  <script src=
  "https://nette.github.io/resources/js/3/netteForms.min.js">
  </script>
  <script src=
  "https://code.jquery.com/jquery-3.5.1.slim.min.js"
  integrity=
  "sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+I
bbVYUew+OrCXaRkfj"
  crossorigin="anonymous"></script>
  <script src="{$basePath}/js/bootstrap.bundle.min.js">
  </script>
  {/block}
</body>
</html>
```

# B Výpis prvního skriptu openssl.py

Výpis B.1: Výpis prvního skriptu openssl.py.

```
#!/usr/bin/env python3
import os, argparse, io

def test_protocols(serverport):
    protocols =  ['ssl2', 'ssl3', 'tls1', 'tls1_1', 'tls1_2']
    # Uložení do ciphers všechny kryptografické sady
    # nástroje openssl
    ciphers = os.popen("openssl␣ciphers␣ALL:eNULL␣|␣\
tr␣:␣'␣'").read().split('␣')
    out = ''
    # Cyklus pro spouštění příkazu s konkrétními protokoly
    # a kryptografickými sadami
    for p in protocols:
        for c in ciphers:
            out += os.popen('openssl␣s_client␣-connect␣\
'+serverport+'␣-cipher␣'+c+'␣-'+p+'␣<␣/dev/null␣>␣/dev/null␣\
2>&1␣&&␣echo␣*'+p+':'+c+',').read()
    bf1 = io.StringIO(out)
    line1 = bf1.readline()

    tls1_3 = "TLS_AES_256_GCM_SHA384␣\
TLS_CHACHA20_POLY1305_SHA256␣TLS_AES_128_GCM_SHA256"
    ciphersTls13 = tls1_3.split('␣')
    out = ''
    # Cyklus pro spouštění příkazu pro TLS 1.3
    # a kryptografickými sadami
    for c in ciphersTls13:
        out += os.popen('openssl␣s_client␣-connect␣\
'+serverport+'␣-ciphersuites␣'+c+'␣-tls1_3␣<␣/dev/null␣>␣\
/dev/null␣2>&1␣&&␣echo␣*tls1_3:'+c+',').read()
    bf2 = io.StringIO(out)
    line2 = bf2.readline()
    suits = ''

    # Označený řádek hvězdičkou (*) je uložen do suits, který
    # se po cyklu vypíše
    while line1:
        if ord('*') == ord(line1[0]):
```

```
            line1 = line1[:0] + '' + line1[0 + 1:]
            suits += line1
            line1 = bf1.readline()
        else:
            line1 = bf1.readline()

    while line2:
        if ord('*') == ord(line2[0]):
            line2 = line2[:0] + '' + line2[0 + 1:]
            suits += line2
            line2 = bf2.readline()
        else:
            line2 = bf2.readline()
    print(suits)

def main():
    parser = argparse.ArgumentParser("Need␣server␣with␣port")
    parser.add_argument("serverport", type=str)
    args = parser.parse_args()
    test_protocols(args.serverport)

if __name__ == '__main__':
    main()
```

# C Výpisy presenteru a modelu pro první skript.

Výpis C.1: Presenter ProtocolsCiphersManager.php v modulu CoreModule.

```
<?php

declare(strict_types=1);

namespace App\CoreModule\Presenters;

use Nette;
use Nette\Application\UI\Form;
use Nette\Application\UI\Presenter;
use App\CoreModule\Models\ProtocolsCiphersManager;

class ProtocolsCiphersPresenter extends Presenter {

    private $protocolsCiphersManager;
    private $output;

    // Konstruktor
    public function __construct(ProtocolsCiphersManager
    $protocolsCiphersManager)
    {
        $this->protocolsCiphersManager =
        $protocolsCiphersManager;
    }

    public function renderDefault(): void {
        $this->template->output = $this->output;
    }

    // Vytvoření formuláře pro zadání serveru
    protected function createComponentServerForm(): Form {
    $form = new Form;
    // Renderer upravuje vzhled formuláře podle Bootstrap 4
        $renderer = $form->getRenderer();
        $renderer->wrappers['controls']['container'] = null;
      $renderer->wrappers['pair']['container'] =
      'div␣class="form-group"';
      $renderer->wrappers['control']['container'] =
      'div␣class="col-sm-9"';
```

```
        $form->addText('server', '')
            ->setRequired()->setAttribute('size', 40);
        $form->addSubmit('send', 'Run');
        $form->onSuccess[] = [$this, 'formSucceeded'];
        return $form;
    }

    public function formSucceeded(Form $form, $data): void {
    // tady zpracujeme data odeslaná formulářem
    // $data->server obsahuje název serveru
        $this->output = $this->protocolsCiphersManager->
        runTest($data->server);
    }
}
```

Výpis C.2: Metoda renderDefault() v presenteru.

```
...
public function renderDefault(): void {
    $this->template->protocols =
    $this->protocolsCiphersManager->protocols;
    $this->template->prot_count =
    $this->protocolsCiphersManager->prot_count;
    $this->template->arrayP =
    $this->protocolsCiphersManager->arrayP;
    $this->template->openssl_version =
    $this->protocolsCiphersManager->openssl_version;
    $this->template->dic_prot = $this->dic_prot;
    $this->template->success = $this->success;
    }
...
```

Výpis C.3: Pomocné proměnné v presenteru ProtocolsCiphersPresenter.

```
...
  // Slovník protokolů
  private $dic_prot = ["ssl2" => "SSLv2",
            "ssl3" => "SSLv3",
            "tls1" => "TLSv1.0",
            "tls1_1" => "TLSv1.1",
            "tls1_2" => "TLSv1.2",
            "tls1_3" => "TLSv1.3"];
```

```
  private $success = false; // výchozí hodnota
...
```

Výpis C.4: Metoda fill_protocols_and_ciphers_to_arrays() v modelu.

```
...
private function fill_protocols_and_ciphers_to_arrays
(string $out_new): void {
        $lines = explode(",", $out_new);
        array_pop($lines);
        $i = 0;
        foreach ($lines as $line) {
            $temp = array();
            $p = explode(":", $line);
            $temp[$p[0]] = $p[1];
            if (in_array($p[0], $this->protocols)){
                array_push($this->arrayP, $temp);
            } else {
                array_push($this->protocols, $p[0]);
                array_push($this->arrayP, $temp);
                $i++;
            }
        }

        // Spočítá se, kolik ciphersuits má konkrétní protokol
        $c = 0;
        foreach ($this->protocols as $prot) {
            foreach ($this->arrayP as $array) {
                if (array_key_exists($prot, $array)) {
                    $c++;
                }
            }
            $this->prot_count[$prot] = $c;
            $c = 0;
        }
    }
...
```

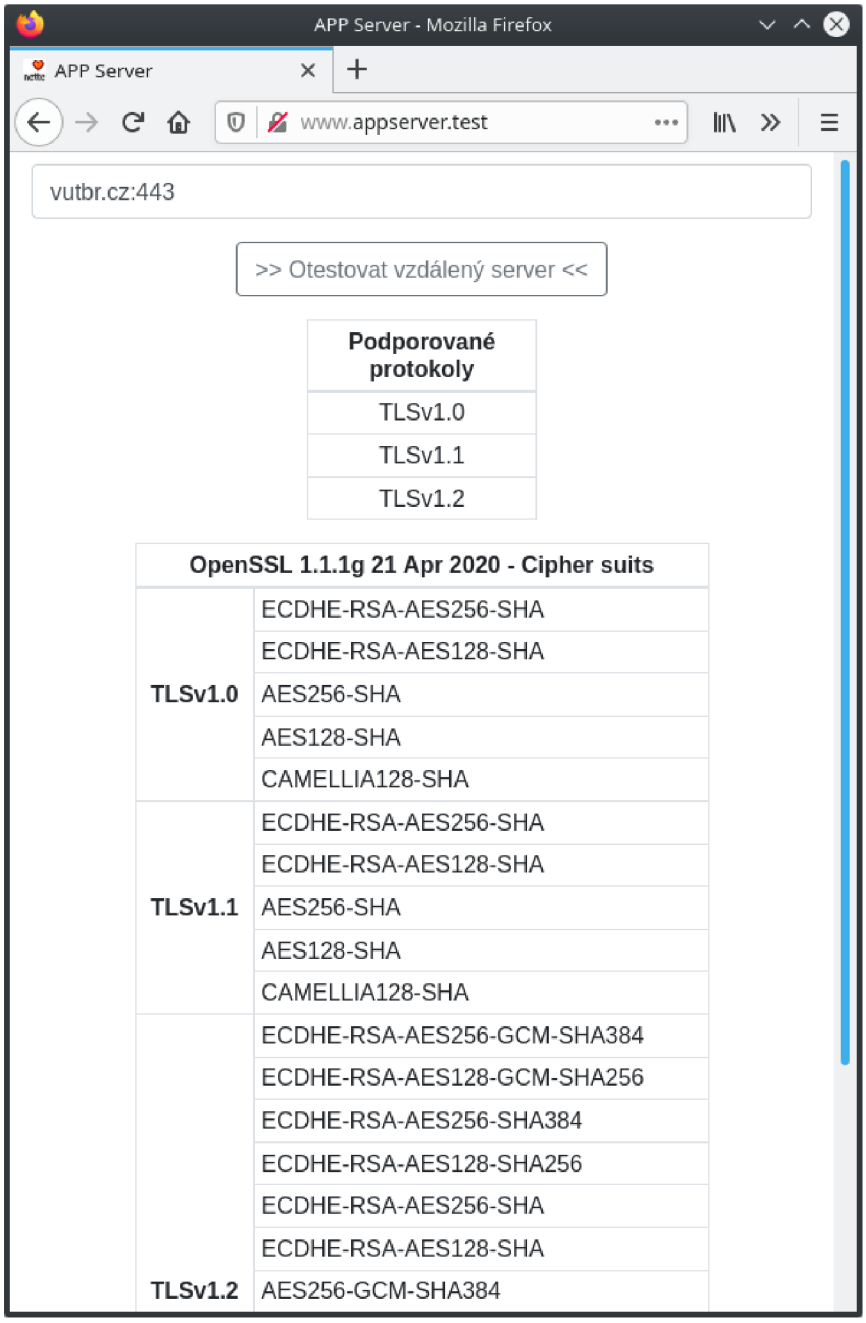
# D Šablona modulu Core

Výpis D.1: Šablona pro zobrazení prvního skriptu.

```
...
<div class="container-fluid">
  <div class="row">
    <div class="col-md-3␣mx-auto">
    {control serverForm}
    </div>
  </div>
  {if $success}
  <div class="row">
    <div class="col-2␣mx-auto">
      <table class="table␣table-bordered␣table-sm">
        <thead>
          <tr>
            <th class="text-center">Podporované protokoly</th>
          </tr>
        </thead>
        <tbody>
          {foreach $protocols as $prot}
            <tr>
              <td class="text-center">{$dic_prot[$prot]}</td>
            </tr>
          {/foreach}
        </tbody>
      </table>
    </div>
  </div>
  <div class="row">
    <div class="col-4␣mx-auto">
      <table class="table␣table-bordered␣table-sm">
        <thead>
          <tr>
            <th colspan=2 class="text-center">
            {$openssl_version} - Cipher suits</th>
          </tr>
        </thead>
        <tbody>
          {foreach $protocols as $prot}
            {var $first = true}
```

```
            {for $k = 0; $k < count($arrayP); $k++}
              {var $array_ciphers = array()}
              {if array_key_exists($prot, $arrayP[$k])}
                {if array_push($array_ciphers,
                $arrayP[$k][$prot])}
                {/if}
              {/if}
              {foreach $array_ciphers as $cipher}
                {if $first}
                  <tr>
                    <th rowspan="{$prot_count[$prot]}"
                    class="text-center␣align-middle">
                    {$dic_prot[$prot]}</th>
                    <td>{$cipher}</td>
                  </tr>
                  {$first = false}
                {else}
                  <tr>
                    <td>{$cipher}</td>
                  </tr>
                {/if}
              {/foreach}
            {/for}
          {/foreach}
        </tbody>
      </table>
    </div>
  </div>
  {/if}
</div>
...
```

# E Ukázka reálného testu prvního skriptu




Obr. E.1: Obrázek reálného testu prvního skriptu na server vutbr.cz.

# F Výpisy implementací dalších skriptů

Výpis F.1: Výpis skriptu nmapinfo.py.

```
#!/usr/bin/env python3
import os, argparse, io

def nmap_info(server):
    ports = '21,22,23,53,80,113,443,3389'
    out = ''
    # Zjištění informací o serveru a sken vybraných portů
    out += os.popen('nmap -p'+ports+' '+server+' --script \
whois-ip').read()
    bf = io.StringIO(out)
    line = bf.readline()

    info = ''
    alterIp = False
    # Dokud existuje řádek
    while line:
        dns = "Nmap scan report for "
        ips = "Other addresses "
        porttext = "PORT"
        who = "Host script results:"
        # Podmínka pro přečtění IP a DNS názvu
        if dns in line:
            # Rozdělí se řetězec jednou a uloží se část
            # pole[1]
            info += line.split(dns,1)[1]
        # Podmínka pro přečtení ostatních IP serveru
        elif ips in line:
            alterIp = True
            aips = line.split(": ",1)[1]
            info += aips.replace(' ', ';')
        elif porttext in line:
            portlist = ''
            # Jestliže server neobsahuje alternativni
            # IP adresy
            if not alterIp: info += '\n'
            line = bf.readline()
            while line:
                # Podmínka pro přečtení informací
```

```
                # o poskytovateli serveru
                if who in line:
                    line = bf.readline()
                    info += portlist.replace('\n', ';')
                    whoinfo = ''
                    info += '\n'
                    while line:
                        line = line.replace("| ","",1) \
                        .replace("|_","",1)
                        whoinfo += line
                        line = bf.readline()
                    info += whoinfo.replace('\n', ';')
                else:
                    portlist += line
                    line = bf.readline()
        else:
            line = bf.readline()
            continue
        line = bf.readline()
    print(info)

def main():
    parser = argparse.ArgumentParser("Need target")
    parser.add_argument("server", type=str)
    args = parser.parse_args()
    nmap_info(args.server)

if __name__ == '__main__':
    main()
```

Výpis F.2: Výpis skriptu certinfo.py.

```
#!/usr/bin/env python3
import os, argparse, io

def cert_info(server):
    # Zjištění informací o certifikátu nástrojem Openssl
    out = ''
    out += os.popen('echo | openssl s_client -connect '\
+server+':443').read()
    bf = io.StringIO(out)
    line = bf.readline()
```

```
    info = ''
    while line:
        if "subject" in line:
            info += line
        elif "issuer" in line:
            info += line
        elif "Secure Renegotiation" in line:
            info += line
        elif "Compression" in line:
            info += line
        line = bf.readline()
    # Zjištění informací o certifikátu nástrojem Nmap
    out = ''
    out += os.popen('nmap -p 443 --script ssl-cert '+server)\
.read()
    bf = io.StringIO(out)
    line = bf.readline()

    while line:
        if "Public Key type" in line:
            info += line.replace("| ","",1)
        elif "Public Key bits" in line:
            info += line.replace("| ","",1)
        elif "Signature Algorithm" in line:
            info += line.replace("| ","",1)
        elif "Not valid before" in line:
            info += line.replace("| ","",1)
        elif "Not valid after" in line:
            info += line.replace("| ","",1)
        line = bf.readline()
    print(info)

def main():
    parser = argparse.ArgumentParser("Need target")
    parser.add_argument("server", type=str)
    args = parser.parse_args()
    cert_info(args.server)

if __name__ == '__main__':
    main()
```

Výpis F.3: Výpis metody get_info_about_server().

```
private function get_info_about_server($out): bool {
    $lines = explode("\n", $out);
    // Odstraní prázdný poslední prvek v poli
    array_pop($lines);

    // Uloží DNS název s IP adresou
    $this->dns = $lines[0];
    $ips = $lines[1];
    if (empty($ips)) $ips = "Žádné alternativní adresy";
    $this->ips = explode(";", $ips);

    // Uloží porty a označí je
    $ports = $lines[2];
    $ports = explode(";", $ports);
    array_pop($ports);
    array_pop($ports);
    $this->ports_only = $ports;
    foreach ($ports as $p) {
        if (strpos($p, "80") !== false &&
        strpos($p, "open") !== false) {
            $this->score += -3;
            $this->port_labels[$p] = "danger";
            $this->set_vulnerability("HTTP");
        } elseif (strpos($p, "21") !== false
        && strpos($p, "open") !== false) {
            $this->score += -5;
            $this->port_labels[$p] = "danger";
            $this->set_vulnerability("21");
        } elseif (strpos($p, "22") !== false
        && strpos($p, "open") !== false) {
            $this->score += -10;
            $this->port_labels[$p] = "danger";
            $this->set_vulnerability("22");
        } elseif (strpos($p, "23") !== false
        && strpos($p, "open") !== false) {
            $this->score += -10;
            $this->port_labels[$p] = "danger";
            $this->set_vulnerability("23");
        } elseif (strpos($p, "113") !== false
        && strpos($p, "open") !== false) {
```

```
            $this->score += -5;
            $this->port_labels[$p] = "danger";
            $this->set_vulnerability("113");
        } elseif (strpos($p, "3389") !== false
        && strpos($p, "open") !== false) {
            $this->score += -10;
            $this->port_labels[$p] = "danger";
            $this->set_vulnerability("3389");
        } elseif (strpos($p, "443") !== false
        && strpos($p, "open") !== false) {
            $this->score += 50;
            $this->port_labels[$p] = "success";
            $this->https = true;
        } elseif (strpos($p, "443") !== false
        && strpos($p, "open") !== true) {
            $this->port_labels[$p] = "danger";
            $this->set_vulnerability("HTTPS");
        } else {
            $this->port_labels[$p] = "success";
        }
    }

    // Uloží informace o poskytovateli
    $infos = $lines[3];
    $this->infos = explode(";", $infos);
    // Odstraní se poslední nepotřebné řádky
    $this->infos = array_filter($this->infos);
    array_pop($this->infos);

    return $this->https;
}
```

Výpis F.4: Výpis metody run_cert_info().

```
private function get_cert_info(string $out): void {
    $lines = explode("\n", $out);
    // Odstraní prázdné hodnoty z pole
    $lines = array_filter($lines);
    // Vyplní informace o certifikátu do proměnných
    foreach ($lines as $line) {
        if (strpos($line, "subject") !== false) {
            $tmp = preg_replace('/subject=/', '$2', $line);
```

```
            $subjectInfo =
            preg_split('/, (?=(?:[^\"]*\"[^\"]*\")*[^\"]*$)/',
                $tmp);
            foreach ($subjectInfo as $i) {
                array_push($this->subject, $i);
            }
        } elseif (strpos($line, "issuer") !== false) {
            $tmp = preg_replace('/issuer=/', '$2', $line);
            $issuerInfo =
            preg_split('/, (?=(?:[^\"]*\"[^\"]*\")*[^\"]*$)/',
                $tmp);
            foreach ($issuerInfo as $i) {
                array_push($this->issuer, $i);
            }
        } elseif (strpos($line, "Renegotiation") !== false) {
            if (preg_match('/IS NOT/', $line)) {
                $this->renegotiation = FALSE;
                $this->set_vulnerability("Renegotiation");
            } else {
                $this->renegotiation = TRUE;
                $this->score += 5;
            }
        } elseif (strpos($line, "Compression") !== false) {
            $this->compression =
            preg_replace('/Compression: /', '$2', $line);
            if (preg_match('/NONE/', $line)) {
                $this->crime = FALSE;
                $this->compression =
                    $this->compression . " (žádná)";
                $this->compression_label[$this->compression] =
                    "success";
            } else {
                $this->crime = TRUE;
                $this->compression_label[$this->compression] =
                    "danger";
                $this->set_vulnerability("CRIME");
                $this->score -= 5;
            }
        } elseif (strpos($line, "Public Key type") !== false)
        {
            $tmp = preg_replace('/Public Key type: /', '$2',
```

```
            $line);
        $this->keyType = strtoupper($tmp);
    } elseif (strpos($line, "Public␣Key␣bits") !== false)
    {
        $tmp = preg_replace('/Public␣Key␣bits:␣/', '$2',
            $line);
        if ($this->keyType == "RSA") {
            $this->keyBits = $tmp;
            if ($tmp >= 2048) {
                $this->keyBits_label[$tmp] = "success";
            } else {
                $this->keyBits_label[$tmp] = "danger";
                $this->score += -10;
                $this->set_vulnerability("Slabý␣klíč");
            }
        } elseif ($this->keyType == "EC") {
            $this->keyBits = $tmp;
            if ($tmp >= 256) {
                $this->keyBits_label[$tmp] = "success";
            } else {
                $this->keyBits_label[$tmp] = "danger";
                $this->score += -10;
                $this->set_vulnerability("Slabý␣klíč");
            }
        }
    } elseif (strpos($line, "Signature␣Algorithm") !==
    false) {
        $tmp = preg_replace('/Signature␣Algorithm:␣/',
            '$2', $line);
        if (preg_match('/\Asha1/', $tmp)) {
            $this->signature_label[$tmp] = "danger";
            $this->score += -5;
            $this->set_vulnerability("SHA1");
        }
        else $this->signature_label[$tmp] = "success";
        $this->signature = $tmp;
    } elseif (strpos($line, "Not␣valid␣before") !== false)
    {
        $tmp = preg_replace('/Not␣valid␣before:␣/', '$2',
            $line);
        $this->validFrom = preg_replace('/T/', '$2␣',
```

```
                $tmp);
        } elseif (strpos($line, "Not␣valid␣after") !== false)
        {
            $tmp = preg_replace('/Not␣valid␣after:␣/', '$2',
                $line);
            $this->validTo = preg_replace('/T/', '$2␣', $tmp);
        }
    }
    // Kontrola platnosti certifikátu
    if (strtotime($this->validFrom) < strtotime($this->
    timeNow) && strtotime($this->timeNow) <
    strtotime($this->validTo)) {
        $this->isValid = TRUE;
    } else {
        $this->isValid = FALSE;
        $this->score -= 10;
        $this->set_vulnerability("Neplatný␣certifikát");
    }
}
```

Výpis F.5: Výpis upravené metody fill_protocols_and_ciphers_to_arrays().

```
private function fill_protocols_and_ciphers_to_arrays
(string $out_new): void {
    $json = file_get_contents("../bin/cipher_mapping.json");
    $this->cipherMapping = Json::decode($json,
        Json::FORCE_ARRAY);
    $lines = explode(",", $out_new);
    array_pop($lines);
    $i = 0;
    // Vyplní pole používaných protokolů a pole protokolů se
    // ciphersuits
    foreach ($lines as $line) {
        $temp = array();
        $p = explode(":", $line);
        // Přepíše název openssl ciphersuits na IANA název,
        // pokud existuje
        if (array_search($p[1], $this->cipherMapping)){
            $p[1] = array_search($p[1], $this->cipherMapping);
        } else {
            continue;
        }
```

```
$temp[$p[0]] = $p[1];
if (in_array($p[0], $this->protocols)){
    array_push($this->arrayP, $temp);
} else {
    array_push($this->protocols, $p[0]);
    array_push($this->arrayP, $temp);
    $i++;
}

// Označí ciphersuits podle podmínek
$this->cipher_labels[$p[1]] = "success";
if (preg_match('/CBC/', $p[1])) {
    $this->cipher_labels[$p[1]] = "warning";
    $this->set_vulnerability("LUCKY13");
}
if (preg_match('/SHA\z/', $p[1])) {
    $this->cipher_labels[$p[1]] = "warning";
    $this->set_vulnerability("SHA1");
}
if (preg_match('/\ATLS_RSA/', $p[1])) {
    $this->cipher_labels[$p[1]] = "warning";
    $this->set_vulnerability("Výměna␣klíče␣přes␣RSA");
}
if (preg_match('/\ATLS_DH/', $p[1])) {
    $this->cipher_labels[$p[1]] = "warning";
    $this->set_vulnerability("RACCOON␣Útok");
}
if (preg_match('/\ATLS_DHE/', $p[1])) {
    $this->cipher_labels[$p[1]] = "warning";
    $this->set_vulnerability("RACCOON␣Útok");
}
if (preg_match('/EXPORT/', $p[1])) {
    $this->cipher_labels[$p[1]] = "danger";
    $this->set_vulnerability("Logjam␣Útok");
}
if (preg_match('/RC2/', $p[1])) {
    $this->cipher_labels[$p[1]] = "danger";
    $this->set_vulnerability("RC");
}
if (preg_match('/RC4/', $p[1])) {
    $this->cipher_labels[$p[1]] = "danger";
```

```
            $this->set_vulnerability("RC");
        }
        if (preg_match('/MD5/', $p[1])) {
            $this->cipher_labels[$p[1]] = "danger";
            $this->set_vulnerability("MD5");
        }
        if (preg_match('/NULL/', $p[1])) {
            $this->cipher_labels[$p[1]] = "danger";
            $this->set_vulnerability("NULL");
        }
        if (preg_match('/DES/', $p[1])) {
            $this->cipher_labels[$p[1]] = "danger";
            $this->set_vulnerability("DES");
        }
        if (preg_match('/DSS/', $p[1])) {
            $this->cipher_labels[$p[1]] = "danger";
            $this->set_vulnerability("DSS");
        }
        if (preg_match('/3DES/', $p[1])) {
            $this->cipher_labels[$p[1]] = "danger";
            $this->set_vulnerability("DES");
        }
    }
    $this->protocols = array_reverse($this->protocols, true);

    // Spočítá se, kolik ciphersuits má konkrétní protokol
    // a označí používané protokoly
    $c = 0;
    foreach ($this->protocols as $prot) {
        foreach ($this->arrayP as $array) {
            if (array_key_exists($prot, $array)) {
                $c++;
            }
        }
        $this->prot_count[$prot] = $c;
        $c = 0;

        if ($prot == "ssl2") {
            $this->score += -15;
            $this->prot_labels[$prot] = "danger";
            $this->set_vulnerability("BEAST␣Útok");
```

```
            $this->set_vulnerability("DROWN␣Útok");
        } elseif ($prot == "ssl3") {
            $this->score += -15;
            $this->prot_labels[$prot] = "danger";
            $this->set_vulnerability("BEAST␣Útok");
            $this->set_vulnerability("POODLE␣Útok");
        } elseif ($prot == "tls1") {
            $this->score += -10;
            $this->prot_labels[$prot] = "danger";
            $this->set_vulnerability("BEAST␣Útok");
        } elseif ($prot == "tls1_1") {
            $this->score += -10;
            $this->prot_labels[$prot] = "danger";
        } elseif ($prot == "tls1_2") {
            $this->score += 10;
            $this->prot_labels[$prot] = "success";
        } elseif ($prot == "tls1_3") {
            $this->score += 20;
            $this->prot_labels[$prot] = "success";
        } else {
            $this->prot_labels[$prot] = "success";
        }
    }
}
```

# G Výpisy oblastí v šabloně default.latte.

Výpis G.1: Základní struktura šablony default.latte podle návrhu.

```
<title>{block title}APP Server{/block}</title>
{block styles}
  <link rel="stylesheet" href="{$basePath}/css/bootstrap.css">
  <link rel="stylesheet" href="{$basePath}/css/style.css">
{/block}
{block content}
<div class="container">
  <div class="row">
    <div class="col-lg-6␣mx-auto">
    {control serverForm}
    </div>
  </div>
  {if $success}
  {if !$online}
    <p style="text-align:center;">Server nenalezen.
    Zkuste to znovu nebo později.</p>
  {else}
  <div class="row">
    <div class="col␣mx-auto">
    <!-- Čas a datum testování -->}
    </div>
    <div class="col␣mx-auto">
    <!-- Legenda -->
    </div>
  </div>
  <div class="row">
    <div class="col␣mx-auto">
      <div class="row">
        <div class="col␣mx-auto">
        <!-- Informace o certifikátu -->
        </div>
      </div>
    </div>
    <div class="col␣mx-auto">
      <div class="row">
        <div class="col␣mx-auto">
        <!-- IP adresy -->
        </div>
```

```
        <div class="col␣mx-auto">
        <!-- Známka a skóre-->
        </div>
      </div>
      <div class="row">
        <div class="col␣mx-auto">
        <!-- Porty -->
        </div>
        <div class="col␣col-xl-4␣mx-auto">
        <!-- Podpora HSTS -->
        <!-- Podporované protokoly -->
        </div>
      </div>
    </div>
  </div>
  <div class="row">
    <div class="col␣mx-auto">
    <!-- Informace o serveru -->
    <!-- Zranitelnosti a doporučení -->
    </div>
    <div class="col␣mx-auto">
    <!-- Šifrovací sady -->
    </div>
  </div>
  {/if}
  {/if}
</div>
{/block}
```

Výpis G.2: Výpis HTML oblasti Čas a datum testování.

```
<table class="table␣table-bordered␣table-sm">
<thead>
  <tr>
    <th colspan=2 class="align-middle␣text-center">
    Informace o testu</th>
  </tr>
</thead>
<tbody>
  <tr>
    <th class="w-50␣align-middle">Zadané URL serveru:</th>
    <td class="align-middle">{$url}</td>
```

```
  </tr>
  <tr>
    <th class="align-middle">Datum a čas testování:</th>
    <td class="align-middle">{$timeNow}</td>
  </tr>
</tbody>
</table>
```

Výpis G.3: Výpis HTML oblasti Legenda.

```
<table class="table␣table-bordered␣table-sm">
<thead>
  <tr>
    <th colspan=4 class="align-middle␣text-center">Legenda
    </th>
  </tr>
</thead>
<tbody>
  <tr>
    <td class="align-middle">Dobře zabezpečený</td>
    <td class="align-middle␣text-center␣text-success">
    A+ (100-95)</td>
    <td class="align-middle">Částečně zabezpečený</td>
    <td class="align-middle␣text-center␣text-warning">
    B (79-50)</td>
  </tr>
  <tr>
    <td class="align-middle">Zabezpečený</td>
    <td class="align-middle␣text-center␣text-success">
    A (94-80)</td>
    <td class="align-middle">Zranitelný</td>
    <td class="align-middle␣text-center␣text-danger">
    C (49-0)</td>
  </tr>
</tbody>
</table>
```

Výpis G.4: Výpis HTML oblasti Informace o certifikátu.

```
<table class="table␣table-bordered␣table-sm">
<thead>
  <tr>
    <th colspan=2 class="text-center">Informace o certifikátu
```

```
    </th>
  </tr>
</thead>
<tbody>
{if $https}
  {var $first = true}
  {foreach $subject as $s}
    {if $first}
      <tr>
        <th rowspan={count($subject)} class="text-center
align-middle">Subjekt</th>
        <td class="align-middle" style="font-size:␣80%;">{$s}
        </td>
      </tr>
      {$first = false}
    {else}
      <tr>
        <td class="align-middle" style="font-size:␣80%;">{$s}
        </td>
      </tr>
    {/if}
  {/foreach}
  {var $first = true}
  {foreach $issuer as $i}
    {if $first}
      <tr>
        <th rowspan={count($issuer)} class="text-center
align-middle">Vydavatel</th>
        <td class="align-middle" style="font-size:␣80%;">{$i}
        </td>
      </tr>
    {$first = false}
    {else}
      <tr>
        <td class="align-middle" style="font-size:␣80%;">{$i}
        </td>
      </tr>
    {/if}
  {/foreach}
    <tr>
      <th class="text-center␣align-middle">Secure
```

```
      <br>Renegotiation</th>
      {if $renegotiation}
        <td class="text-center␣align-middle">Bezpečné znovu
        vyjednávání <span class="text-success">podporuje
        </span></td>
      {else}
        <td class="text-center␣align-middle">Bezpečné znovu
        vyjednávání <span class="text-danger">nepodporuje
        </span></td>
      {/if}
    </tr>
    <tr>
      <th class="text-center␣align-middle">Komprese</th>
        <td class="text-center␣align-middle
text-{$compression_label[$compression]}">
        {$compression}</td>
    </tr>
    <tr>
      <th class="text-center␣align-middle">Veřejný klíč</th>
      <td class="text-center␣align-middle
text-{$keyBits_label[$keyBits]}">{$keyType} {$keyBits} bitů
      </td>
    </tr>
    <tr>
      <th class="text-center␣align-middle">Podpis</th>
      <td class="text-center␣align-middle
text-{$signature_label[$signature]}">{$signature}</td>
    </tr>
    <tr>
      <th class="text-center␣align-middle">Platnost od</th>
      <td class="text-center␣align-middle">{$validFrom}</td>
    </tr>
    <tr>
      <th class="text-center␣align-middle">Platnost do</th>
      <td class="text-center␣align-middle">{$validTo}</td>
    </tr>
    <tr>
      {if $isValid}
        <th colspan=2 class="text-center␣align-middle
text-success">Certifikát je platný</th>
      {else}
```

```
        <th colspan=2 class="text-center␣align-middle
text-danger">Certifikát je neplatný</th>
      {/if}
    </tr>
{else}
  <tr>
    <td class="text-center">Žádné informace o certifikátu</td>
  </tr>
{/if}
</tbody>
</table>
```

Výpis G.5: Výpis HTML oblasti IP adresy.

```
<table class="table␣table-bordered␣table-sm">
<thead>
  <tr>
    <th class="text-center">IP adresy</th>
  </tr>
</thead>
<tbody>
  <tr>
    <td>{$dns}</td>
  </tr>
  {foreach $ips as $ip}
    <tr>
      <td style="font-size:␣80%;">{$ip}</td>
    </tr>
  {/foreach}
</tbody>
</table>
```

Výpis G.6: Výpis HTML oblasti Známka a skóre.

```
<table class="table␣table-bordered␣table-sm">
<thead>
  <tr>
    <th class="text-center␣align-middle">Známka</th>
    <th class="text-center␣align-middle">Skóre serveru</th>
  </tr>
</thead>
<tbody>
  <tr>
```

```
  {if $score < 0}
  <td class="text-center␣display-3␣text-danger">C</td>
    <td class="text-center␣display-3␣text-danger">0</td>
  {elseif $score < 50}
    <td class="text-center␣display-3␣text-danger">C</td>
    <td class="text-center␣display-3␣text-danger">{$score}
    </td>
  {elseif $score < 80}
    <td class="text-center␣display-3␣text-warning">B</td>
    <td class="text-center␣display-3␣text-warning">{$score}
    </td>
  {else}
    {if $score < 95}
      <td class="text-center␣display-3␣text-success">A</td>
    {else}
      <td class="text-center␣display-3␣text-success">A+</td>
    {/if}
    <td class="text-center␣display-3␣text-success">{$score}
    </td>
  {/if}
  </tr>
</tbody>
</table>
```

Výpis G.7: Výpis HTML oblasti Porty.

```
<table class="table␣table-bordered␣table-sm">
<thead>
  <tr>
    <th class="text-center">Porty</th>
  </tr>
</thead>
<tbody>
  {foreach $ports_only as $port}
    <tr>
      <td class="text-{$port_labels[$port]}">{$port}</td>
    </tr>
  {/foreach}
</tbody>
</table>
```

Výpis G.8: Výpis HTML oblasti Podpora HSTS.

```
<table class="table␣table-bordered␣table-sm">
<thead>
  <tr>
    <th class="text-center">Podpora HSTS</th>
  </tr>
</thead>
<tbody>
  {if $isHsts}
    <tr>
      <td class="text-center␣text-success">Ano</td>
    </tr>
  {else}
    <tr>
      <td class="text-center␣text-danger">Ne</td>
    </tr>
  {/if}
</tbody>
</table>
```

Výpis G.9: Výpis HTML oblasti Podporované protokoly.

```
<table class="table␣table-bordered␣table-sm">
<thead>
  <tr>
    <th class="text-center">Podporované protokoly</th>
  </tr>
</thead>
<tbody>
  {if $https}
    {foreach $protocols as $prot}
      <tr>
        <td class="text-center␣text-{$prot_labels[$prot]}">
        {$dic_prot[$prot]}</td>
      </tr>
    {/foreach}
  {else}
    <tr>
      <td class="text-center">Žádné</td>
    </tr>
  {/if}
</tbody>
</table>
```

Výpis G.10: Výpis HTML oblasti Informace o serveru.

```
<table class="table␣table-bordered␣table-sm">
<thead>
  <tr>
    <th class="text-center">Informace o serveru</th>
  </tr>
</thead>
<tbody>
  {foreach $infos as $info}
    <tr>
      <td>{$info}</td>
    </tr>
  {/foreach}
</tbody>
</table>
```

Výpis G.11: Výpis HTML oblasti Zranitelnosti a doporučení.

```
<table class="table␣table-bordered␣table-sm">
<thead>
  <tr>
    <th class="text-center␣w-25">Zranitelnosti</th>
    <th class="text-center">Doporučení</th>
  </tr>
</thead>
<tbody>
  {ifset $vulnerabilities}
  {foreach $vulnerabilities as $v}
    <tr>
      <td class="align-middle␣text-center␣text-danger">{$v}
      </td>
      <td class="align-middle" style="font-size:␣90%;">
        <div class="text-center">
        {$vulnMapping[$v]['description']}</div>
        <div class="text-center">
        <a href="{$vulnMapping[$v]['html']}" target="_blank">
        {$vulnMapping[$v]['htmlText']}</a></div>
      </td>
    </tr>
  {/foreach}
  {else}
    <td colspan=2 class="␣text-center␣text-success">Žádné
```

```
    zranitelnosti nebyly detekovány.</td>
  {/ifset}
</tbody>
</table>
```

Výpis G.12: Výpis HTML oblasti Šifrovací sady.

```
<table class="table␣table-bordered␣table-sm">
<thead>
  <tr>
    <th colspan=2 class="text-center">
    {$openssl_version} - Cipher suits (IANA)</th>
  </tr>
</thead>
<tbody>
{if $https}
  {foreach $protocols as $prot}
    {var $first = true}
    {for $k = 0; $k < count($arrayP); $k++}
      {var $array_ciphers = array()}
      {if array_key_exists($prot, $arrayP[$k])}
        {if array_push($array_ciphers, $arrayP[$k][$prot])}
        {/if}
      {/if}
      {foreach $array_ciphers as $cipher}
        {if $first}
          <tr>
            <th rowspan="{$prot_count[$prot]}"
            class="text-center␣align-middle">
            {$dic_prot[$prot]}</th>
            <td class="text-{$cipher_labels[$cipher]}"
            style="font-size:␣75%;">{$cipher}</td>
          </tr>
          {$first = false}
        {else}
          <tr>
            <td class="text-{$cipher_labels[$cipher]}"
            style="font-size:␣75%;">{$cipher}</td>
          </tr>
        {/if}
      {/foreach}
    {/for}
```

```
  {/foreach}
{else}
  <tr>
    <td class="text-center">Žádné informace o šifrovacích sad
    </td>
  </tr>
{/if}
</tbody>
</table>
```

# H Ukázka webových stránek s výsledky

vutbr.cz

>> Otestovat vzdálený server <<

| Informace o testu | |
|---|---|
| **Zadané URL serveru:** | vutbr.cz |
| **Datum a čas testování:** | 2021-05-20 10:32:07 |

| Legenda | | | |
|---|---|---|---|
| Dobře zabezpečený | A+ (100-95) | Částečně zabezpečený | B (79-50) |
| Zabezpečený | A (94-80) | Zranitelný | C (49-0) |

| Informace o certifikátu | |
|---|---|
| **Subjekt** | C = CZ |
| | L = Brno-st\C5\99ed |
| | O = Vysok\C3\A9 u\C4\8Den\C3\AD technick\C3\A9 v Brn\C4\9B |
| | CN = www.vutbr.cz |
| **Vydavatel** | C = NL |
| | O = GEANT Vereniging |
| | CN = GEANT OV RSA CA 4 |
| **Secure Renegotiation** | Bezpečné znovu vyjednávání podporuje |
| **Komprese** | NONE (žádná) |
| **Veřejný klíč** | RSA 2048 bitů |
| **Podpis** | sha384WithRSAEncryption |
| **Platnost od** | 2021-04-26 00:00:00 |
| **Platnost do** | 2022-04-26 23:59:59 |
| Certifikát je platný | |

| IP adresy |
|---|
| vutbr.cz (147.229.2.90) |
| Žádné alternativní adresy |

| Známka | Skóre serveru |
|---|---|
| C | 42 |

| Porty |
|---|
| 21/tcp filtered ftp |
| 22/tcp filtered ssh |
| 23/tcp filtered telnet |
| 53/tcp filtered domain |
| 80/tcp open http |
| 113/tcp filtered ident |
| 443/tcp open https |
| 3389/tcp filtered ms-wbt-server |

| Podpora HSTS |
|---|
| Ne |

| Podporované protokoly |
|---|
| TLSv1.2 |
| TLSv1.1 |
| TLSv1.0 |

| Informace o serveru |
|---|
| whois-ip: Record found at whois.ripe.net |
| inetnum: 147.229.0.0 - 147.229.254.255 |
| netname: VUTBRNET |
| descr: Brno University of Technology |
| country: CZ |
| role: Brno University of Technology - Backbone Admins |
| email: admin@cis.vutbr.cz |

| Zranitelnosti | Doporučení |
|---|---|
| HTTP | Server má povolený nezabezpečený port 80. Hrozí potenciální odposlech spojení. Jak přesměrovat na HTTPS (Apache) |
| HSTS | Nedetekováno - potenciální útok MITM. Konfigurace HSTS |
| LUCKY13 | Byl použit mód CBC při šifrování. CVE-2013-0169 Detail (NIST) |
| SHA1 | Byl použit slabý hash SHA1. Prolomený SHA1 (NIST) |
| Výměna klíče přes RSA | Výměna klíče proběhla pomocí RSA algoritmu. Doporučení NIST pro TLS |
| BEAST Útok | Byly použity protokoly TLS 1.0 a starší. Doporučení NIST pro TLS |

| OpenSSL 1.1.1d 10 Sep 2019 - Cipher suits (IANA) | |
|---|---|
| **TLSv1.2** | TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 |
| | TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 |
| | TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 |
| | TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 |
| | TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA |
| | TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA |
| | TLS_RSA_WITH_AES_256_GCM_SHA384 |
| | TLS_RSA_WITH_AES_128_GCM_SHA256 |
| | TLS_RSA_WITH_AES_256_CBC_SHA256 |
| | TLS_RSA_WITH_AES_128_CBC_SHA256 |
| | TLS_RSA_WITH_AES_256_CBC_SHA |
| | TLS_RSA_WITH_AES_128_CBC_SHA |
| | TLS_RSA_WITH_CAMELLIA_128_CBC_SHA |
| **TLSv1.1** | TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA |
| | TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA |
| | TLS_RSA_WITH_AES_256_CBC_SHA |
| | TLS_RSA_WITH_AES_128_CBC_SHA |
| | TLS_RSA_WITH_CAMELLIA_128_CBC_SHA |
| **TLSv1.0** | TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA |
| | TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA |
| | TLS_RSA_WITH_AES_256_CBC_SHA |
| | TLS_RSA_WITH_AES_128_CBC_SHA |
| | TLS_RSA_WITH_CAMELLIA_128_CBC_SHA |

Obr. H.1: Ukázka webové stránky s výsledky serveru vutbr.cz.

webserver-actinver-prd.lfr.cloud

>> Otestovat vzdálený server <<

| Informace o testu | |
|---|---|
| **Zadané URL serveru:** | webserver-actinver-prd.lfr.cloud |
| **Datum a čas testování:** | 2021-05-20 10:16:05 |

| Legenda | | | |
|---|---|---|---|
| Dobře zabezpečený | A+ (100-95) | Částečně zabezpečený | B (79-50) |
| Zabezpečený | A (94-80) | Zranitelný | C (49-0) |

| Informace o certifikátu | |
|---|---|
| **Subjekt** | C = US |
| | postalCode = 91765 |
| | ST = California |
| | L = Diamond Bar |
| | street = 1400 Montefino Ave. |
| | O = "Liferay, Inc." |
| | CN = liferay.cloud |
| **Vydavatel** | C = GB |
| | ST = Greater Manchester |
| | L = Salford |
| | O = Sectigo Limited |
| | CN = Sectigo RSA Organization Validation Secure Server CA |
| **Secure Renegotiation** | Bezpečné znovu vyjednávání nepodporuje |
| **Komprese** | NONE (žádná) |
| **Veřejný klíč** | RSA 2048 bitů |
| **Podpis** | sha256WithRSAEncryption |
| **Platnost od** | 2020-05-20 00:00:00 |
| **Platnost do** | 2021-10-24 23:59:59 |
| **Certifikát je platný** | |

| IP adresy |
|---|
| webserver-actinver-prd.lfr.cloud (35.186.243.200) |
| Žádné alternativní adresy |

| Známka | Skóre serveru |
|---|---|
| A | 82 |

| Porty |
|---|
| 21/tcp filtered ftp |
| 22/tcp filtered ssh |
| 23/tcp filtered telnet |
| 53/tcp filtered domain |
| 80/tcp open http |
| 113/tcp filtered ident |
| 443/tcp open https |
| 3389/tcp open ms-wbt-server |

| Podpora HSTS |
|---|
| Ano |

| Podporované protokoly |
|---|
| TLSv1.3 |
| TLSv1.2 |

| Informace o serveru |
|---|
| whois-ip: Record found at whois.arin.net |
| netrange: 35.184.0.0 - 35.191.255.255 |
| netname: GOOGLE-CLOUD |
| orgname: Google LLC |
| orgid: GOOGL-2 |
| country: US stateprov: CA |
| orgtechname: Google LLC |
| orgtechemail: arin-contact@google.com |

| OpenSSL 1.1.1d 10 Sep 2019 - Cipher suits (IANA) | |
|---|---|
| **TLSv1.3** | TLS_AES_256_GCM_SHA384 |
| | TLS_CHACHA20_POLY1305_SHA256 |
| | TLS_AES_128_GCM_SHA256 |
| **TLSv1.2** | TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 |
| | TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 |
| | TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 |
| | TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA |
| | TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA |

| Zranitelnosti | Doporučení |
|---|---|
| HTTP | Server má povolený nezabezpečený port 80. Hrozí potenciální odposlech spojení. Jak přesměrovat na HTTPS (Apache) |
| 3389 | Server má otevřený port 3389 (RDP). Otevřený přístup ke vzdálené ploše. Doporučeno port zakázat nebo nastavit jiný pro vzdálený přístup. Informace |
| Renegotiation | Server nepodporuje bezpečné opětovné vyjednávání. Informace |
| LUCKY13 | Byl použit mód CBC při šifrování. CVE-2013-0169 Detail (NIST) |
| SHA1 | Byl použit slabý hash SHA1. Prolomený SHA1 (NIST) |

Obr. H.2: Výsledky serveru webserver-actinver-prd.lfr.cloud.

fekt.vut.cz

>> Otestovat vzdálený server <<

| Informace o testu | |
|---|---|
| **Zadané URL serveru:** | fekt.vut.cz |
| **Datum a čas testování:** | 2021-05-20 10:27:50 |

| Legenda | | | |
|---|---|---|---|
| Dobře zabezpečený | A+ (100-95) | Částečně zabezpečený | B (79-50) |
| Zabezpečený | A (94-80) | Zranitelný | C (49-0) |

| Informace o certifikátu | |
|---|---|
| **Subjekt** | CN = www.fekt.vut.cz |
| **Vydavatel** | C = US |
| | O = Let's Encrypt |
| | CN = R3 |
| **Secure Renegotiation** | Bezpečné znovu vyjednávání nepodporuje |
| **Komprese** | NONE (žádná) |
| **Veřejný klíč** | RSA 2048 bitů |
| **Podpis** | sha256WithRSAEncryption |
| **Platnost od** | 2021-05-05 07:05:55 |
| **Platnost do** | 2021-08-03 07:05:55 |
| **Certifikát je platný** | |

| IP adresy |
|---|
| fekt.vut.cz (147.229.71.28) |
| 2001:67c:1220:9847::93e5:471c |

| Známka | Skóre serveru |
|---|---|
| C | 47 |

| Porty |
|---|
| 21/tcp closed ftp |
| 22/tcp open ssh |
| 23/tcp closed telnet |
| 53/tcp closed domain |
| 80/tcp open http |
| 113/tcp closed ident |
| 443/tcp open https |
| 3389/tcp closed ms-wbt-server |

| Podpora HSTS |
|---|
| Ne |

| Podporované protokoly |
|---|
| TLSv1.3 |
| TLSv1.2 |
| TLSv1.1 |
| TLSv1.0 |

| Informace o serveru |
|---|
| whois-ip: Record found at whois.ripe.net |
| inetnum: 147.229.0.0 - 147.229.254.255 |
| netname: VUTBRNET |
| descr: Brno University of Technology |
| country: CZ |
| role: Brno University of Technology - Backbone Admins |
| email: admin@cis.vutbr.cz |

| Zranitelnosti | Doporučení |
|---|---|
| 22 | Server má otevřený port 22 (SSH). Potenciální útok uhádnutím hesla. Doporučeno port zakázat nebo nastavit jiný pro SSH přístup. |
| HTTP | Server má povolený nezabezpečený port 80. Hrozí potenciální odposlech spojení. Jak přesměrovat na HTTPS (Apache) |
| Renegotiation | Server nepodporuje bezpečné opětovné vyjednávání. Informace |
| HSTS | Nedetekováno - potenciální útok MITM. Konfigurace HSTS |
| LUCKY13 | Byl použit mód CBC při šifrování. CVE-2013-0169 Detail (NIST) |
| SHA1 | Byl použit slabý hash SHA1. Prolomený SHA1 (NIST) |
| RACCOON Útok | Výměna kliče proběhla pomocí DH nebo DHE algoritmu. Informace |
| Výměna klíče přes RSA | Výměna kliče proběhla pomocí RSA algoritmu. Doporučení NIST pro TLS |
| BEAST Útok | Byly použitý protokoly TLS 1.0 a starší. Doporučení NIST pro TLS |

| OpenSSL 1.1.1d 10 Sep 2019 - Cipher suits (IANA) | |
|---|---|
| **TLSv1.3** | TLS_AES_256_GCM_SHA384 |
| | TLS_CHACHA20_POLY1305_SHA256 |
| | TLS_AES_128_GCM_SHA256 |
| **TLSv1.2** | TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 |
| | TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 |
| | TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 |
| | TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 |
| | TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 |
| | TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 |
| | TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 |
| | TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 |
| | TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 |
| | TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA |
| | TLS_DHE_RSA_WITH_AES_256_CBC_SHA |
| | TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA |
| | TLS_DHE_RSA_WITH_AES_128_CBC_SHA |
| | TLS_RSA_WITH_AES_256_GCM_SHA384 |
| | TLS_RSA_WITH_AES_128_GCM_SHA256 |
| | TLS_RSA_WITH_AES_256_CBC_SHA256 |
| | TLS_RSA_WITH_AES_128_CBC_SHA256 |
| | TLS_RSA_WITH_AES_256_CBC_SHA |
| | TLS_RSA_WITH_AES_128_CBC_SHA |
| **TLSv1.1** | TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA |
| | TLS_DHE_RSA_WITH_AES_256_CBC_SHA |
| | TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA |
| | TLS_DHE_RSA_WITH_AES_128_CBC_SHA |
| | TLS_RSA_WITH_AES_256_CBC_SHA |
| | TLS_RSA_WITH_AES_128_CBC_SHA |
| **TLSv1.0** | TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA |
| | TLS_DHE_RSA_WITH_AES_256_CBC_SHA |
| | TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA |
| | TLS_DHE_RSA_WITH_AES_128_CBC_SHA |
| | TLS_RSA_WITH_AES_256_CBC_SHA |
| | TLS_RSA_WITH_AES_128_CBC_SHA |

Obr. H.3: Výsledky serveru fekt.vut.cz.