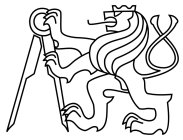
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Automatické testování bezpečného nastavení služeb se šifrovanými protokoly
Student:	Bc. Martin Volek
Vedoucí:	Mgr. Rudolf Bohumil Blažek, Ph.D.
Studijní program:	Informatika
Studijní obor:	Počítačová bezpečnost
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Prostudujte běžně používané zabezpečené komunikační protokoly včetně protokolů služeb VPN, protokol SSH a rodinu protokolů SSL/TLS pro služby SMTPS, IMAPS, POP3S a HTTPS. Prostudujte metody navázání zabezpečeného spojení těmito protokoly a analyzujte související bezpečnostní rizika. Prostudujte pravidla pro nastavení serverů a klientů zaručující navázání bezpečné komunikace. Navrhněte metodologii pro testování tohoto nastavení. Implementujte nástroj pro testování dodržování těchto pravidel pro vybrané protokoly. V implementaci se zaměřte především na služby jiné než HTTPS. Prostudujte vhodnost implementace nástroje jako zásuvného modulu pro penetrační testovací software nmap. Otestujte vytvořený nástroj na vlastním počítači a ve virtuální počítačové síti či v bourací laboratoři FIT ČVUT v Praze.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

prof. Ing. Róbert Lórencz, CSc.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 14. prosince 2015

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

Diplomová práce

Automatické testování bezpečného nastavení služeb se šifrovanými protokoly

Bc. Martin Volek

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

10. ledna 2017

Poděkování

Děkuji dr. Rudolfu B. Blažkovi, vedoucímu této práce, za připomínky a rady, kterými přispěl k vypracování této diplomové práce.

Prohlášení

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

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

V Praze dne 10. ledna 2017 .....

Odkaz na tuto práci

Volek, Martin. *Automatické testování bezpečného nastavení služeb se šifrovanými protokoly.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato práce se zabývá kontrolou bezpečné konfigurace služeb s vybranými šifrovanými síťovými protokoly. Součástí práce je studie kryptografických technologií běžně používaných v síťové komunikaci včetně bezpečnostních aspektů jejich konfigurace. V práci je navrhnut a implementován nástroj pro automatizované testování bezpečné konfigurace vybraných šifrovaných síťových protokolů a souvisejících služeb. Nástroj byl otestován v reálném síťovém prostředí a je využitelný v praxi.

Klíčová slova síťová bezpečnost, šifrovaný síťový protokol, kontrola konfigurace, navázání spojení, ověření identity, šifra, ssl, tls, ssh, vpn

Abstract

This thesis deals with verifying secure configuration of services with selected encrypted network protocols. The thesis includes a study of cryptographic technologies commonly used in network communications, including security aspects of their configuration. A part of the thesis is the design and implemetation of a testing tool for automated verification of secure configuration of selected encrypted network protocols and the corresponding services. The tool has been tested in a real network environment and is usable in real life.

Keywords network security, encrypted network protocol, configuration verification, identity verification, handshake, cipher, ssl, tls, ssh, vpn

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V dnešní době je převážná většina komunikace a přenosu informací uskutečněna elektronickou cestou. To přináší problém bezpečnosti a důvěryhodnosti takové komunikace, protože informace v elektronické podobě je velmi jednoduše zfalšovatelná. Na internetu lze najít spoustu návodů, díky kterým může kdokoli provádět různé kybernetické útoky, přičemž k tomu nepotřebuje hluboké znalosti dané problematiky, protože obsahují jednoduché návody krok po kroku [1]. Je tedy nutné zavádět a používat různé mechanismy, které nám poskytnou možnosti jak komunikovat s jistou úrovní bezpečnosti, abychom mohli přenášeným informacím důvěřovat. Při používání těchto bezpečnostních mechanismů je kvalita a důvěryhodnost závislá na správném nastavení. Kontrolou tohoto správného nastavení se právě zabývá tato práce.

Cílem této práce je popsat často používané zabezpečené síťové protokoly a analyzovat aspekty jejich konfigurace, které jsou důležité pro zaručení bezpečnosti. Konkrétně se jedná o protokoly HTTPS, SSH, emailové protokoly a VPN protokoly.

Zejména je v práci provedena studie metod navázání spojení (handshake) pro jednotlivé protokoly a analýza jejich bezpečnostních rizik, které mohou být způsobeny nevhodnou konfigurací.

Dále je cílem práce navrhnout a implementovat nástroj, který bude automatizovaně testovat bezpečné nastavení vybraných zabezpečených (šifrovaných) protokolů a služeb, které je využívají. Tento nástroj bude fungovat tak, že se nakonfiguruje pro testování konkrétních služeb se šifrovanými protokoly. Toto testování bude možné spouštět jednorázově s okamžitým výstupem. Také bude možné testování spouštět automatizovaně v pravidelných intervalech a v případě výskytu bezpečnostního problému automaticky informovat správce.

Kapitola 1 popisuje výsledky analýzy běžně používaných šifrovaných protokolů, souvisejících služeb a jejich konfigurace. Návrh vytvořeného nástroje je popsán v kapitole 2 a jeho implementace je popsána v kapitole 3. Kapitola 4 shrnuje výsledky testování implementovaného nástroje.

Kapitola 1

Analýza

Problém síťové komunikace, tak jak vznikala od svých počátků, spočívá v nízké bezpečnosti základních verzí protokolů [2]. Základními verzemi je zde myšlena verze bez podpory ověřování identity, autorizace a bez šifrování. Tyto protokoly uskutečňují veškerou komunikaci v otevřené (nešifrované) textové či binární formě. To znamená, že pokud tato komunikace využívá nějaký kanál, který není pod stoprocentní kontrolou, je tato komunikace z principu nedůvěryhodná. U internetu i u naprosté většiny sítí nikdy nelze mít kanál stoprocentně pod kontrolou už ze samotného principu fungování sítí, protože komunikace probíhá přes uzly (poskytovatele), které nemáme možnost ovládat [3]. U nedůvěryhodné komunikace se nelze spolehnout na to, že:

- komunikace není odposlouchávána,
- komunikace není po cestě záměrně měněna,
- komunikace pochází z ověřeného zdroje.

Typickým útokem na takovouto nezabezpečenou komunikaci je tzv. **Man-in-the-Middle** (MitM, MiM) útok [4, 5]. Tento útok spočívá v tom, že se útočník postaví mezi dva komunikující subjekty a komunikaci odposlouchává, případně pozměňuje. MitM útok lze použít i na zabezpečenou komunikaci, tam je ale odposlouchávání a případně padělání zpráv výrazně ztíženo nebo při správném použití znemožněno [6].

K tomu, aby byla komunikace bezpečná, je nutné zajistit:

- aby komunikace nebyla odposlouchávána,
- aby komunikace nebyla po cestě záměrně měněna,
- aby komunikace probíhala mezi ověřenými subjekty.

Bezpečná komunikace probíhá ve dvou krocích:

- navázání komunikace (tzv. handshake) včetně ověření identity,
- následně šifrovaná komunikace.

1.1 Šifrování obecně

Při každém šifrování je nutné mít nějaký klíč k šifrování a k dešifrování. Tyto klíče musí zůstat tajné, přičemž jimi musí disponovat obě komunikující strany (v případě asymetrické šifry obě strany disponují jinou částí klíče). Jsou tedy dvě možnosti, první je mít klíče předem domluvené, zde je slabina správa klíčů - je nutné zajistit, aby klíč nebyl odcizen či jinak kompromitován. Druhou možností je se na klíči domluvit před samotnou komunikací, zde slabina spočívá v ověření identity komunikujících strana a v samotné výměně klíče.

1.1.1 Symetrické a asymetrické šifry

Základním dělením šifrovacích algoritmů je dělení na symetrické a asymetrické šifry [7]. U symetrických šifer je šifrovací a dešifrovací klíč stejný. Asymetrické šifry mají dva různé klíče, jeden pro šifrování a jeden pro dešifrování. Důležité je, že ze znalosti jednoho klíče nelze získat klíč druhý (alespoň jedním směrem).

1.1.2 Využití asymetrických šifer při šifrování

Asymetrické šifry bývají výpočetně více náročné oproti symetrickým šifrám [8, 9, 10]. Z tohoto důvodu se asymetrické šifry většinou nepoužívají na šifrování přenášených dat. Pomocí asynchronní šifry si strany obvykle vymění šifrovací klíč k synchronní šifře a pomocí této synchronní šifry se pak šifrují data. Tato počáteční navázání spojení se nazývá handshake. Dále se při handshaku pomocí asynchronních šifer ověřuje identita komunikujících stran.

1.1.3 Handshake

K tomu, aby mohla probíhat komunikace šifrovaně, je třeba, aby si strany vyměnily šifrovací klíč a navzájem ověřili identitu druhé strany. Tento proces se nazývá handshake. Handshake je nejkritičtější místo bezpečnosti celé komunikace. Je totiž třeba si společný klíč vyměnit tak, aby nebyl odposlechnut či podvrhnut, přičemž tato výměna je započata nezabezpečeným kanálem. K tomuto se používají různé metody, například algoritmy RSA key exchange [11, 12, 7] nebo Diffie–Hellman [13, 7], které jsou popsány v sekci 1.1.7.

1.1.4 Ověření identity během handshaku

Aby byl handshake důvěryhodný, je nutné ověřit, zda opravdu komunikujeme s tím subjektem, se kterým si myslíme. Mohlo by se totiž stát, že máme sice velmi dobře zabezpečenou komunikaci, ale komunikujeme přímo s útočníkem.

Často se při komunikaci ověřuje identita serveru pomocí certifikátu identity nebo veřejného klíče. Klient se obvykle autentizuje kombinací uživatelského jména a hesla. Pro bezpečnější ověření identity klienta lze ale také použít certifikát či jeho veřejný klíč.

Ověření důvěryhodnosti pomocí certifikátu

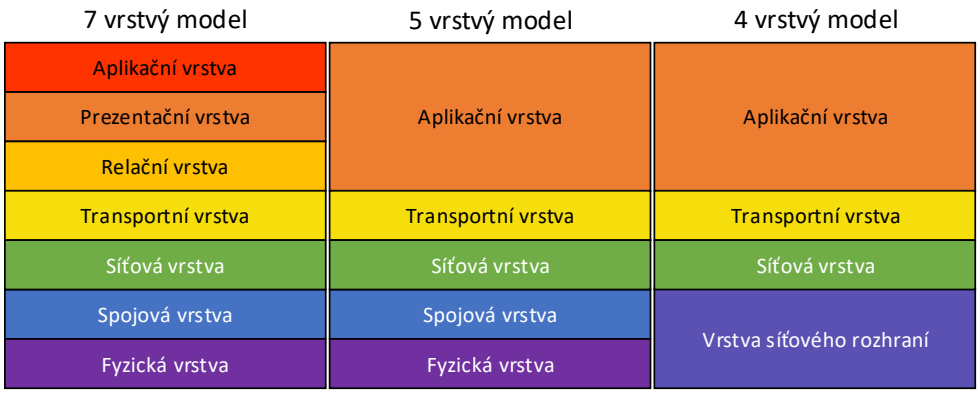
Důvěryhodnost certifikátu identity zajišťuje jeho digitální podpis. Důvěryhodný certifikát znamená certifikát podepsaný certifikační autoritou, které věříme. Obvykle je seznam certifikátů důvěryhodných certifikačních autorit přímo v operačním systému (případně je součástí software, kterým komunikujeme), je ale možné do něj zasahovat. Těmto certifikátům se říká kořenové certifikáty, protože stojí na vrcholu řetězce důvěry, kterým se iterativně ověřuje autenticita certifikátu serveru a případných intermediate autorit [14, 15].

Pokud máme obavu o důvěryhodnost těchto kořenových certifikačních autorit, můžeme si vytvořit vlastní certifikační autoritu a pomocí ní podepisovat vlastní certifikáty. Je nutné pak ale certifikát naší certifikační autority rozdistribuovat mezi všechna zařízení a prohlásit jej za důvěryhodný. Je třeba ale myslet na to, že distribuce certifikátu naší certifikační autority musí jít zabezpečenou cestou, což může být problém, pokud nemáme ke všem zařízením bezpečný či fyzický přístup.

Pokud tedy máme důvěryhodný certifikát identity komunikačního partnera, můžeme pomocí veřejného klíče v tomto certifikátu zašifrovat data, aby šla rozšifrovat jen příslušným soukromým klíčem, kterým by měla disponovat jen strana se kterou chceme komunikovat (např. server). Více o asymetrických šifrách v sekci 1.1.1.

1.1.5 Šifrovaná komunikace

Za předpokladu, že šifrovaná komunikace probíhá s opravdu tajnými klíči (znají je jen komunikující strany), je kompromitování komunikace nepravděpodobné. Záleží samozřejmě na kvalitě šifry, ale při použití kvalitní šifry se v dnešní době není třeba obávat prolomení bez znalosti klíčů [16]. Útok půjde pravděpodobně směrem k odhalení klíčů nějakou postranní metodou, například napadením handshaku, spíše než snahou rozšifrovat zprávu jen se znalostí šifrovaného textu [17]. Co je a není kvalitní šifra se můžeme rozhodnout na základě doporučení národních či soukromých organizací, které se touto proble-



Obrázek 1.1: Modely sítí

matikou zaobírají. V dnešní době je za kvalitní šifrovací algoritmus považováno zejména AES[1] [18].

1.1.6 Šifrování na síťových vrstvách

Bezpečnost komunikace (šifrování) může být prováděno na různých vrstvách síťového modelu. V této práci se budeme věnovat především šifrování na síťové a aplikační vrstvě.

Referenční síťové modely

Existuje několik referenčních síťových modelů. Na obrázku 1.1 je diagram následujících síťových modelů:

- 7 vrstvý model (ISO[2]/OSI[3]) - ISO/IEC[4] 7498-1:1994 [19],
- 4 vrstvý model (TCP/IP[5]) - RFC[6] 1122 [20], RFC 1123 [21],
- 5 vrstvý model (někdy používán jako alternativa k TCP/IP modelu).

Šifrování na síťové vrstvě

Hlavní výhoda šifrování na síťové vrstvě je ta, že není potřeba přímá podpora šifrování pro aplikace pracující na vrstvě vyšší než síťové. Šifrování je tedy transparentní a je možné zabezpečit komunikace aplikací, které samy o sobě

[1] Advanced Encryption Standard
[2] International Organization for Standardization
[3] Open Systems Interconnection model
[4] International Electrotechnical Commission
[5] Transmission Control Protocol/Internet Protocol
[6] Request for Comments

šifrování nepodporují. K šifrování na síťové vrstvě se používá protokol IPsec[7] [22], který je popsán v sekci 1.4.3.

Šifrování na aplikační vrstvě

Šifrování na aplikační (či prezentační) vrstvě vyžaduje přímou podporu aplikacemi. Každá aplikace zde může použít svůj vlastní způsob šifrování. Je ale zvykem použít nějaký ověřený a funkční bezpečnostní protokol a nad ním postavit protokol s vlastní (již bezpečnou) komunikací.

Například služby jako HTTPS[8], IMAP[9], SMTP[10], POP3[11] a OpenVPN používají pro šifrování rodinu protokolů SSL[12]/TLS[13].

1.1.7 Konkrétní šifry

Příklady symetrických šifer

- AES - Advanced Encryption Standard,
- DES (3DES) - (Triple) Data Encryption Standard,
- Blowfish.

Příklady asymetrických šifer

- RSA - Rivest, Shamir & Adleman,
- Diffie-Hellman,
- ElGamal,
- DSA - Digital Signature Algorithm.

Jak již bylo zmíněno, nejkritičtější místo celé komunikace je handshake. Při handhaku se právě používají asymetrické šifry. Z tohoto důvodu se budeme v této práci věnovat především asymetrickým šifrám.

[7] Internet Protocol Security
[8] Hypertext Transfer Protocol Secure
[9] Internet Message Access Protocol
[10] Simple Mail Transfer Protocol
[11] Post Office Protocol 3
[12] Secure Sockets Layer
[13] Transport Layer Security

RSA

RSA je asymetrická šifra, která umožňuje jak šifrování, tak elektronické podepisování [11]. RSA je založena na tom, že existuje pár klíčů - soukromý klíč a veřejný klíč.

Šifrování pomocí RSA funguje tak, že data zašifrovaná veřejným klíčem lze rozšifrovat jen soukromým klíčem. Strana, která chce šifrovat, má tedy k dispozici veřejný klíč. Pomocí tohoto veřejného klíče zašifruje požadovaná data. Tato zašifrovaná data je pak schopný rozšifrovat jen subjekt, který disponuje soukromým klíčem. Tato data tedy lze poslat nezabezpečeným kanálem a lze se spolehnout na to, že se k datům nedostane nikdo jiný než vlastník soukromého klíče.

Tato myšlenka je ovšem platná pouze za předpokladu, že nedojde k podvržení veřejného klíče. Pokud by k takovému podvržení došlo, tak lze provést útok Man-in-the-Middle. Veřejné klíče jsou ale často distribuovány jako součást certifikátu o jehož důvěryhodnosti lze rozhodnout na základě podpisu důvěryhodných certifikačních autorit. Jinak je třeba veřejné klíče předat jiným bezpečným způsobem, např. na CD či podobném fyzickém mediu.

Toto šifrování funguje i opačně, lze šifrovat soukromým klíčem a dešifrovat veřejným klíčem. Tímto způsobem se ověřuje autenticita zpráv (podpis).

Diffie-Hellman

Diffie-Hellman je algoritmus, který umožňuje dvěma a více stranám bezpečnou výměnu tajného šifrovacího klíče. Klíč získaný pomocí tohoto algoritmu nelze zpětně rekonstruovat ani v případě, že by případný útočník odchytil veškerou komunikaci, protože část klíče se nikam neukládá ani nikam neposílá [13].

Pravdivost tvrzení o útočníkově nemožnosti rekonstrukce šifrovacího klíče získaného pomocí algoritmu Diffie–Hellman je založena na předpokladu, že neexistuje algoritmus, který by tento problém vyřešil v polynomiálním čase. Takový algoritmus dosud není znám, není ovšem dokázáno, že neexistuje. Dosud nejlepší známý algoritmus dosahuje kvazi-polynomialního času [23].

Diffie-Hellman pro dvě strany (Alice, Bob) [13, 24]

1. strany se dohodnou na veřejných parametrech g a prvočíslu p, přičemž g je generátor grupy $G = (Z_p^*; \cdot)$,
2. Alice si zvolí tajný parametr a a spočítá $x = g^a \bmod p$ a pošle Bobovi,
3. Bob si zvolí tajný parametr b a spočítá $y = g^b \bmod p$ a pošle Alici,
4. Alice spočítá $k = y^a \bmod p = (g^b \bmod p)^a \bmod p = g^{ba} \bmod p$,
5. Bob spočítá $k = x^b \bmod p = (g^a \bmod p)^b \bmod p = g^{ab} \bmod p$,
6. v tuto chvíli mají obě strany klíč $k = g^{ab} \bmod p = g^{ba} \bmod p$.

Takto pomocí algoritmu Diffie-Hellman dojde k výměně tajného klíče mezi dvěma stranami, přičemž veškerá tato komunikace jde nezabezpečeným kanálem. Algoritmus lze jednoduše upravit tak, aby fungoval mezi více stranami.

Dále je důležité poznamenat, že v příkladě výše uvedeném je použit algoritmus Diffie-Hellman, který je založen na umocňování v multiplikativní grupě nad přirozenými čísly. Jeho bezpečnost je tedy určena obtížností vyřešit (diskrétní) odmocninu v této multiplikativní grupě. Lze ale použít i jiné než multiplikativní grupy. Další možností je použít například aditivní grupu nad body eliptické křivky [25, 26].

1.1.8 Zákon o kybernetické bezpečnosti

Od 1. ledna 2015 je v České republice účinný zákon o kybernetické bezpečnosti č. 181/2014 Sb. Tento zákon upravuje práva a povinnosti osob a působnost a pravomoci orgánů veřejné moci v oblasti kybernetické bezpečnosti [27]. Zákon je povinný pouze pro orgány veřejné moci. Ustanovení tohoto zákona lze však využít jako doporučení pro komerční nebo soukromou sféru.

Dále je vhodné brát v potaz doporučení i jiných organizací, např. NIST (National Institute of Standards and Technology) a jeho FIPS (Federal Information Processing Standards).

Prováděcí předpisy k zákonu o kybernetické bezpečnosti

- Vládní nařízení č. 315/2014 o odvětvových kritériích pro určení prvku kritické infrastruktury,
- vyhláška č. 316/2014 Sb. o kybernetické bezpečnosti,
- vyhláška č. 317/2014 Sb. o významných informačních systémech a jejich určujících kritériích.

Minimální požadavky na kryptografické algoritmy

Minimální požadavky na kryptografické algoritmy jsou určeny v příloze 3 vyhlášky 316/2014 Sb. Seznam těchto požadavků je ve trochu zjednodušené formě vypsán dále. Kompletní seznam je přímo v příloze vyhlášky [28].

- Symetrické algoritmy:
  - Blokové a proudové šifry pro ochranu důvěrnosti a integrity:
    - AES 128, 192, 256,
    - 3DES 112, 168 (omezené použití),
    - Blowfish min. 128 (omezené použití),
    - Kasumi 128 (omezené použití),
    - Twofish 128-256,

        * Serpent 128, 192, 256,
        * Camellia 128, 192, 256,
        * SNOW 2.0, SNOW 3G 128, 256,
    - Módy šifrování s ochranou integrity:
        * CCM,
        * EAX,
        * OCB,
        * Složená schémata typu „Encrypt-then-MAC“,
    - Módy šifrování:
        * CTR,
        * OFB,
        * CBC,
        * CFB,
    - Módy pro ochranu integrity:
        * HMAC,
        * CBC-MAC-X9.19 (omezené použití),
        * CBC-MAC-EMAC,
        * CMAC,
- Asymetrické algoritmy:
    - Pro technologii digitálního podpisu:
        * DSA 2048,
        * EC-DSA 224,
        * RSA-PSS 2048,
    - Pro procesy dohod na klíči a šifrování klíčů:
        * DH 2048,
        * ECDH 224,
        * ECIES-KEM 256,
        * PSEC-KEM 256,
        * ACE-KEM 256,
        * RSA-OAEP 2048,
        * RSA-KEM 2048,
- Algoritmy hash funkcí:
    - SHA1 (omezené použití),
    - SHA2 224, 256, 384, 512, 512/224, 512/256,
    - SHA3 224, 256, 384, 512,

- SHAKE 128, 256,
- Whirpool,
- RIPEMD 160.

1.2 SSL/TLS

V předchozí sekci bylo popsáno obecně, jak se zajišťuje bezpečnost síťové komunikace. Zde se konkrétně zaměřím na protokoly SSL a TLS. Tyto protokoly umožňují zabezpečenou komunikaci pomocí šifrování, ověření identity a autentizace komunikujících stran. Používá se pro zajištění bezpečnosti mnoha existujících protokolů (HTTP, POP3, IMAP, SMTP, OpenVPN atd.). Protokol SSL je starší verzí protokolu TLS.

Jednotlivé verze SSL

- **SSL 2.0** - první veřejná verze, již zakázána v RFC 6176 [29],
- **SSL 3.0** (RFC 6101 [30]) - již zakázána v RFC 7568 [31].

Jednotlivé verze TLS

- **TLS 1.0** (RFC 2246 [32]),
- **TLS 1.1** (RFC 4346 [33]),
- **TLS 1.2** (RFC 5246 [34]),
- **TLS 1.3** - není finální, v současné době je ve formě draftu [35].

SSL se tedy dnes již nedoporučuje používat. Místo toho je vhodné použít protokol TLS v některé jeho verzi, který je považován za bezpečný [31] (při použití správných verzí knihoven, které opravují chyby a známé zranitelnosti).

1.2.1 Verze a zranitelnosti protokolů SSL/TLS

V následujících sekcích se budu věnovat bezpečnostním zranitelnostem a rozdílům jednotlivých verzí protokolů SSL a TLS [36].

SSL 2.0

SSL 2.0 je první veřejnou verzí. Od března 2011 je již její použití zakázáno v RFC 6176 [29]. SSL 2.0 už by tedy neměla být povolena ani jako náhradní metoda (tzv. fallback).

Závažné bezpečnostní problémy SSL 2.0

- SSL 2.0 umožňuje použití slabé šifry se 40 bitovým klíčem,
- SSL 2.0 umožňuje útočníkovi editovat seznam podporovaných šifer, a tak donutit server i klienta k použití slabé šifry,
- SSL 2.0 používá slabou konstrukci MAC[14],
- SSL 2.0 přidává padding k MAC, přičemž délka paddingu není podepsána, to umožňuje útočníkovi mazat konce zpráv.

1.2.1.1 SSL 3.0

SSL 3.0 opravuje některé bezpečností problémy vyskytující se v 2.0. Je také zakázána od června 2015 v RFC 7568 [31]. SSL 3.0 už by tedy neměla být povolena ani jako náhradní (fallback) metoda.

Hlavní výhody oproti SSL 2.0 více viz [37, 38]

- SSL 3.0 používá 128 bitové klíče,
- SSL 3.0 opravuje chybu, kdy byl útočník schopen editovat seznam podporovaných šifer, a tak donutit server i klienta k použití slabé šifry tím, že přidává do handshake hash všech předchozích handshaků,
- SSL 3.0 opravuje slabou konstrukci MAC,
- SSL 3.0 správně podepisuje délku paddingu,
- SSL 3.0 přesouvá zodpovědnost za volbu šifrovacích a kompresních algoritmů na server (ve 2.0 klient),
- SSL 3.0 umožňuje provést nový handhake uprostřed spojení na žádost klienta i serveru,
- SSL 3.0 umožňuje používat hierarchii certifikátů hlubší než 2,
- SSL 3.0 zavádí podporu výměny klíčů pomocí Diffie-Hellman a Fortezza algoritmů,
- SSL 3.0 podporuje non-RSA certifikáty.

Závažné bezpečnostní problémy SSL 3.0

- SSL 3.0 umožňuje pomocí útoku padding oracle zjistit obsah šifrovaných zpráv [39].

[14] Message Authentication Protocol

1.2.1.2 TLS 1.0

TLS 1.0 vychází z SSL 3.0 jen s malými rozdíly. Protokoly ale i tak nejsou navzájem kompatibilní [36].

Hlavní výhody oproti SSL 3.0 [36]

- TLS 1.0 generuje proměnlivou délku paddingu (což stěžuje analýzu délky zprávy),
- TLS 1.0 používá jiné funkce pro generování klíčů,
- TLS 1.0 používá finální HMAC[15],
- TLS 1.0 vyžaduje podporu DSS/DH[16],
- TLS 1.0 nepodporuje výměnu klíčů pomocí algoritmu Fortrezza a symetrického šifrovacího klíče,
- TLS 1.0 zavádí drobné změny ve zprávách a ukončení spojení.

Závažné bezpečnostní problémy TSL 1.0

- TLS 1.0 umožňuje pomocí útoku padding oracle zjistit obsah šifrovaných zpráv (jen v některých implementacích) [40].

1.2.1.3 TLS 1.1

Hlavní výhody oproti TLS 1.0 [36]

- TLS 1.1 zavádí explicitní inicializační vektory (IV) místo implicitních IV (ochrana kvůli útoků na CBC operační mód),
- TLS 1.1 mění decryption_failed alert na bad_record_mac alert při chybě s paddingem (ochrana kvůli útoků na CBC operační mód).

Závažné bezpečnostní problémy TSL 1.1 [36]

- TLS 1.1 umožňuje pomocí útoku padding oracle zjistit obsah šifrovaných zpráv (jen v některých implementacích) [40].

[15] Keyed-Hash Message Authentication Code
[16] Digital Signature Standard/Diffie–Hellman

1.2.1.4 TLS 1.2

Hlavní výhody oproti TSL 1.1 [36]

- TLS 1.2 nahrazuje kombinaci MD5-SHA-1 za SHA-256
- TLS 1.2 zásadně mění možnosti, jak server i klient určuje, které hashovací a podpisové algoritmy podporují.

1.2.2 Podporované algoritmy na výměnu klíčů SSL/TLS

V tabulce 1.1 jsou podporované algoritmy pro výměnu klíčů pro jednotlivé verze SSL/TLS. V pravé části tabulky je znázorněno, které algoritmy jsou podporované v konkrétních verzích protokolu [30, 32, 33, 34, 41].

Jedná se pouze o algoritmy podporované, záleží pak na konfiguraci a implementaci serveru, které podporované algoritmy jsou implementované a povolené. Je důležité při konfiguraci dbát na to, aby byly povoleny jen kvalitní a bezpečné algoritmy.

1.2.3 Podporované šifrovací algoritmy SSL/TLS

V tabulce 1.2 jsou uvedeny podporované šifrovací algoritmy pro jednotlivé verze SSL/TLS [30, 32, 33, 34, 41].

Šifrovací algoritmy v tabulce jsou pouze podporované. Je pak určeno konfigurací a implementací serveru, které šifrovací algortimy jsou opravdu povolené. Je nutné konfigurovat server tak, aby byly použité povolené jen bezpečné šifrovací algoritmy.

1.2.4 Knihovna OpenSSL

Knihovna OpenSSL je open source knihovna, která implementuje protokoly SSL a TLS [42]. Knihovna je napsána v jazyce C a lze ji využívat v různých jazycích a na různých operačních systémech.

OpenSSL nejen implementuje mnoho šifrovacích sad SSL a TLS, které jsou definovány v příslušných RFC dokumentech, ale také poskytuje informace o těchto šifrovacích sadách a každou šifrovací sadu zařazuje do nějaké úrovně bezpečnosti (high, medium a low).

OpenSSL lze tedy použít pro ohodnocení bezpečnosti jednotlivých šifrovacích sad. To samozřejmě za předpokladu, že knihovně OpenSSL a jejím autorům v tomto směru důvěřujeme a používáme aktuální verzi knihovny.

1.2.4.1 Informace a ohodnocení šifrovacích sad v OpenSSL

Informace a ohodnocení šifrovacích sad v OpenSSL lze získat pomocí příkazu `openssl ciphers`.

Tabulka 1.1: Podporované algoritmy na výměnu klíčů SSL/TLS [30, 32, 33, 34, 41]

Algoritmus	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2
RSA	Ano	Ano	Ano	Ano	Ano
DH-RSA	Ne	Ano	Ano	Ano	Ano
DHE-RSA	Ne	Ano	Ano	Ano	Ano
ECDH-RSA	Ne	Ne	Ano	Ano	Ano
ECDHE-RSA	Ne	Ne	Ano	Ano	Ano
DH-DSS	Ne	Ano	Ano	Ano	Ano
DHE-DSS	Ne	Ano	Ano	Ano	Ano
ECDH-ECDSA	Ne	Ne	Ano	Ano	Ano
ECDHE-ECDSA	Ne	Ne	Ano	Ano	Ano
PSK	Ne	Ne	Ano	Ano	Ano
PSK-RSA	Ne	Ne	Ano	Ano	Ano
DHE-PSK	Ne	Ne	Ano	Ano	Ano
ECDHE-PSK	Ne	Ne	Ano	Ano	Ano
SRP	Ne	Ne	Ano	Ano	Ano
SRP-DSS	Ne	Ne	Ano	Ano	Ano
SRP-RSA	Ne	Ne	Ano	Ano	Ano
Kerberos	Ne	Ne	Ano	Ano	Ano
DH-ANON	Ne	Ne	Ano	Ano	Ano
ECDH-ANON	Ne	Ne	Ano	Ano	Ano

Příklady využití příkazu `openssl ciphers`

- `openssl ciphers 'ALL:eNULL@STRENGTH'` - vypsání všech podporovaných šifer,
- `openssl ciphers 'HIGH@STRENGTH'` - vypsání šifer s vysokou úrovní bezpečnosti,
- `openssl ciphers 'MEDIUM@STRENGTH'` - vypsání šifer se střední úrovní bezpečnosti,
- `openssl ciphers 'LOW:eNULL@STRENGTH'` - vypsání šifer s nízkou či žádnou úrovní bezpečnosti,
- s přepínačem `-v` vypíše podrobně jednotlivé části šifrovacích sad (viz sekce 1.2.4.3 a tabulka 1.4).

1.2.4.2 Názvy šifrovacích sad v OpenSSL

Drobný problém u OpenSSL spočívá v tom, že OpenSSL používá jiné vlastní názvy šifrovacích sad oproti RFC dokumentům, ve kterých jsou specifikovány

Tabulka 1.2: Podporované šifrovací algoritmy jednotlivých verzí SSL/TLS [30, 32, 33, 34, 41]

Šifra			**Protokol**				
Typ	**Algoritmus**	**Délka klíče**	**SSL 2.0**	**SSL 3.0**	**TLS 1.0**	**TLS 1.1**	**TLS 1.2**
Bloková šifra	**AES GCM**	256, 128	Ne	Ne	Ne	Ne	Ano
	AES CCM		Ne	Ne	Ne	Ne	Ano
	AES CBC		Ne	Ne	Ano	Ano	Ano
	Camellia GCM	256, 128	Ne	Ne	Ne	Ne	Ano
	Camellia CBC		Ne	Ne	Ano	Ano	Ano
	ARIA GCM	256, 128	Ne	Ne	Ne	Ne	Ano
	ARIA CBC		Ne	Ne	Ano	Ano	Ano
	SEED CBC	128	Ne	Ne	Ano	Ano	Ano
	3DES EDE CBC	112	Ano	Ano	Ano	Ano	Ano
	GOST 28147-89 CNT	256	Ne	Ne	Ano	Ano	Ano
	IDEA CBC	128	Ano	Ano	Ano	Ano	Ne
	DES CBC	56	Ano	Ano	Ano	Ano	Ne
		40	Ano	Ano	Ano	Ne	Ne
	RC2 CBC	40	Ano	Ano	Ano	Ne	Ne
Proudová šifra	**ChaCha20-Poly1305**	256	Ne	Ne	Ne	Ne	Ano
	RC4	128	Ano	Ano	Ano	Ano	Ano
		40	Ano	Ano	Ano	Ne	Ne
Žádná šifra	**-**	-	Ne	Ano	Ano	Ano	Ano

jednotlivé verze protokolů SSL/TLS. Pokud tedy chceme pracovat se správnými RFC názvy, tak musíme názvy překládat. Mapování z OpenSSL názvů na RFC názvy je možné najít přímo v manuálu k OpenSSL, konkrétně v manuálu příkazu „`openssl ciphers`“ [43]. Mapování názvů z OpenSSL do RFC názvů je v tabulce 1.3.

1.2.4.3 Jednotlivé části šifrovací sady

Šifrovací sada má nějaký název, pod kterým se skrývají jednotlivé metody, které se používají pro části šifrované komunikace.

Například `DHE-RSA-AES256-SHA256` představuje následující kombinaci:

- **DH (Diffie-Hellman)** - algoritmus pro výměnu klíčů. Zpravidla se jedná asymetrický šifrovací algoritmus.
- **RSA** - algoritmus pro autentizaci (zde se tedy nepoužívá RSA pro výměnu klíčů i když by to bylo možné). Zde se také zpravidla jedná o asymetrický šifrovací algoritmus.
- **AES 256** - šifrovací algoritmus, kterým se šifruje samotná komunikace po handshaku. Zpravidla se jedná symetrický šifrovací algoritmus, který může být blokový i proudový.
- **SHA 256** - hashovací algoritmus.

Ne vždy je ale z názvu úplně zřejmé, jaké jednotlivé algoritmy představuje nějaký název šifrovací sady. OpenSSL ale tyto informace poskytuje jako součást příkazu „`openssl ciphers`“. Konkrétní kombinace pro jednotlivé šifrovací sady získané z OpenSSL (verze 1.0.2g) jsou uvedeny v tabulce 1.4.

Tabulka 1.3: Mapování názvu šifrovacích sad z RFC do OpenSSL. Barvy znázorňují úroveň bezpečnosti z OpenSSL 1.0.2g (vysoká, střední, nízká).

RFC název	OpenSSL název
SSL_CK_RC4_128_WITH_MD5	RC4-MD5
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA	ADH-DES-CBC3-SHA
SSL_DH_anon_WITH_RC4_128_MD5	ADH-RC4-MD5
SSL_DH_DSS_WITH_3DES_EDE_CBC_SHA	DH-DSS-DES-CBC3-SHA
SSL_DH_RSA_WITH_3DES_EDE_CBC_SHA	DH-RSA-DES-CBC3-SHA
SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA	EDH-DSS-DES-CBC3-SHA
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA	EDH-RSA-DES-CBC3-SHA
SSL_RSA_WITH_3DES_EDE_CBC_SHA	DES-CBC3-SHA
SSL_RSA_WITH_NULL_MD5	NULL-MD5
SSL_RSA_WITH_NULL_SHA	NULL-SHA
SSL_RSA_WITH_RC4_128_MD5	RC4-MD5
SSL_RSA_WITH_RC4_128_SHA	RC4-SHA
TLS_DH_anon_WITH_3DES_EDE_CBC_SHA	ADH-DES-CBC3-SHA
TLS_DH_anon_WITH_AES_128_CBC_SHA	ADH-AES128-SHA
TLS_DH_anon_WITH_AES_128_CBC_SHA256	ADH-AES128-SHA256
TLS_DH_anon_WITH_AES_128_GCM_SHA256	ADH-AES128-GCM-SHA256

Tabulka 1.3: Mapování názvu šifrovacích sad z RFC do OpenSSL. Barvy znázorňují úroveň bezpečnosti z OpenSSL 1.0.2g (vysoká, střední, nízká).

RFC název	OpenSSL název
TLS_DH_anon_WITH_AES_256_CBC_SHA	ADH-AES256-SHA
TLS_DH_anon_WITH_AES_256_CBC_SHA256	ADH-AES256-SHA256
TLS_DH_anon_WITH_AES_256_GCM_SHA384	ADH-AES256-GCM-SHA384
TLS_DH_anon_WITH_CAMELLIA_128_CBC_SHA	ADH-CAMELLIA128-SHA
TLS_DH_anon_WITH_CAMELLIA_256_CBC_SHA	ADH-CAMELLIA256-SHA
TLS_DH_anon_WITH_RC4_128_MD5	ADH-RC4-MD5
TLS_DH_anon_WITH_SEED_CBC_SHA	ADH-SEED-SHA
TLS_DH_DSS_WITH_AES_128_CBC_SHA	DH-DSS-AES128-SHA
TLS_DH_DSS_WITH_AES_128_CBC_SHA256	DH-DSS-AES128-SHA256
TLS_DH_DSS_WITH_AES_128_GCM_SHA256	DH-DSS-AES128-GCM-SHA256
TLS_DH_DSS_WITH_AES_256_CBC_SHA	DH-DSS-AES256-SHA
TLS_DH_DSS_WITH_AES_256_CBC_SHA256	DH-DSS-AES256-SHA256
TLS_DH_DSS_WITH_AES_256_GCM_SHA384	DH-DSS-AES256-GCM-SHA384
TLS_DH_DSS_WITH_CAMELLIA_128_CBC_SHA	DH-DSS-CAMELLIA128-SHA
TLS_DH_DSS_WITH_CAMELLIA_256_CBC_SHA	DH-DSS-CAMELLIA256-SHA
TLS_DH_DSS_WITH_SEED_CBC_SHA	DH-DSS-SEED-SHA
TLS_DH_RSA_WITH_AES_128_CBC_SHA	DH-RSA-AES128-SHA
TLS_DH_RSA_WITH_AES_128_CBC_SHA256	DH-RSA-AES128-SHA256
TLS_DH_RSA_WITH_AES_128_GCM_SHA256	DH-RSA-AES128-GCM-SHA256
TLS_DH_RSA_WITH_AES_256_CBC_SHA	DH-RSA-AES256-SHA
TLS_DH_RSA_WITH_AES_256_CBC_SHA256	DH-RSA-AES256-SHA256
TLS_DH_RSA_WITH_AES_256_GCM_SHA384	DH-RSA-AES256-GCM-SHA384
TLS_DH_RSA_WITH_CAMELLIA_128_CBC_SHA	DH-RSA-CAMELLIA128-SHA
TLS_DH_RSA_WITH_CAMELLIA_256_CBC_SHA	DH-RSA-CAMELLIA256-SHA
TLS_DH_RSA_WITH_SEED_CBC_SHA	DH-RSA-SEED-SHA
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	EDH-DSS-DES-CBC3-SHA
TLS_DHE_DSS_WITH_AES_128_CBC_SHA	DHE-DSS-AES128-SHA
TLS_DHE_DSS_WITH_AES_128_CBC_SHA256	DHE-DSS-AES128-SHA256
TLS_DHE_DSS_WITH_AES_128_GCM_SHA256	DHE-DSS-AES128-GCM-SHA256
TLS_DHE_DSS_WITH_AES_256_CBC_SHA	DHE-DSS-AES256-SHA
TLS_DHE_DSS_WITH_AES_256_CBC_SHA256	DHE-DSS-AES256-SHA256
TLS_DHE_DSS_WITH_AES_256_GCM_SHA384	DHE-DSS-AES256-GCM-SHA384
TLS_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA	DHE-DSS-CAMELLIA128-SHA
TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA	DHE-DSS-CAMELLIA256-SHA
TLS_DHE_DSS_WITH_SEED_CBC_SHA	DHE-DSS-SEED-SHA
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	EDH-RSA-DES-CBC3-SHA
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	DHE-RSA-AES128-SHA
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	DHE-RSA-AES128-SHA256
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	DHE-RSA-AES128-GCM-SHA256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA	DHE-RSA-AES256-SHA
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	DHE-RSA-AES256-SHA256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	DHE-RSA-AES256-GCM-SHA384
TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA	DHE-RSA-CAMELLIA128-SHA
TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA	DHE-RSA-CAMELLIA256-SHA
TLS_DHE_RSA_WITH_SEED_CBC_SHA	DHE-RSA-SEED-SHA
TLS_ECDH_anon_WITH_3DES_EDE_CBC_SHA	AECDH-DES-CBC3-SHA
TLS_ECDH_anon_WITH_AES_128_CBC_SHA	AECDH-AES128-SHA
TLS_ECDH_anon_WITH_AES_256_CBC_SHA	AECDH-AES256-SHA
TLS_ECDH_anon_WITH_NULL_SHA	AECDH-NULL-SHA
TLS_ECDH_anon_WITH_RC4_128_SHA	AECDH-RC4-SHA
TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA	ECDH-ECDSA-DES-CBC3-SHA
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA	ECDH-ECDSA-AES128-SHA
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256	ECDH-ECDSA-AES128-SHA256
TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256	ECDH-ECDSA-AES128-GCM-SHA256
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA	ECDH-ECDSA-AES256-SHA
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384	ECDH-ECDSA-AES256-SHA384

Tabulka 1.3: Mapování názvu šifrovacích sad z RFC do OpenSSL. Barvy znázorňují úroveň bezpečnosti z OpenSSL 1.0.2g (vysoká, střední, nízká).

RFC název	OpenSSL název
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384	ECDH-ECDSA-AES256-GCM-SHA384
TLS_ECDH_ECDSA_WITH_NULL_SHA	ECDH-ECDSA-NULL-SHA
TLS_ECDH_ECDSA_WITH_RC4_128_SHA	ECDH-ECDSA-RC4-SHA
TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA	ECDH-RSA-DES-CBC3-SHA
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA	ECDH-RSA-AES128-SHA
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256	ECDH-RSA-AES128-SHA256
TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256	ECDH-RSA-AES128-GCM-SHA256
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA	ECDH-RSA-AES256-SHA
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384	ECDH-RSA-AES256-SHA384
TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384	ECDH-RSA-AES256-GCM-SHA384
TLS_ECDH_RSA_WITH_NULL_SHA	ECDH-RSA-NULL-SHA
TLS_ECDH_RSA_WITH_RC4_128_SHA	ECDH-RSA-RC4-SHA
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA	ECDHE-ECDSA-DES-CBC3-SHA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	ECDHE-ECDSA-AES128-SHA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	ECDHE-ECDSA-AES128-SHA256
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	ECDHE-ECDSA-AES128-GCM-SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	ECDHE-ECDSA-AES256-SHA
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	ECDHE-ECDSA-AES256-SHA384
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	ECDHE-ECDSA-AES256-GCM-SHA384
TLS_ECDHE_ECDSA_WITH_NULL_SHA	ECDHE-ECDSA-NULL-SHA
TLS_ECDHE_ECDSA_WITH_RC4_128_SHA	ECDHE-ECDSA-RC4-SHA
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	ECDHE-RSA-DES-CBC3-SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	ECDHE-RSA-AES128-SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	ECDHE-RSA-AES128-SHA256
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	ECDHE-RSA-AES128-GCM-SHA256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	ECDHE-RSA-AES256-SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	ECDHE-RSA-AES256-SHA384
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDHE-RSA-AES256-GCM-SHA384
TLS_ECDHE_RSA_WITH_NULL_SHA	ECDHE-RSA-NULL-SHA
TLS_ECDHE_RSA_WITH_RC4_128_SHA	ECDHE-RSA-RC4-SHA
TLS_PSK_WITH_3DES_EDE_CBC_SHA	PSK-3DES-EDE-CBC-SHA
TLS_PSK_WITH_AES_128_CBC_SHA	PSK-AES128-CBC-SHA
TLS_PSK_WITH_AES_256_CBC_SHA	PSK-AES256-CBC-SHA
TLS_PSK_WITH_RC4_128_SHA	PSK-RC4-SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA	DES-CBC3-SHA
TLS_RSA_WITH_AES_128_CBC_SHA	AES128-SHA
TLS_RSA_WITH_AES_128_CBC_SHA256	AES128-SHA256
TLS_RSA_WITH_AES_128_GCM_SHA256	AES128-GCM-SHA256
TLS_RSA_WITH_AES_256_CBC_SHA	AES256-SHA
TLS_RSA_WITH_AES_256_CBC_SHA256	AES256-SHA256
TLS_RSA_WITH_AES_256_GCM_SHA384	AES256-GCM-SHA384
TLS_RSA_WITH_CAMELLIA_128_CBC_SHA	CAMELLIA128-SHA
TLS_RSA_WITH_CAMELLIA_256_CBC_SHA	CAMELLIA256-SHA
TLS_RSA_WITH_NULL_MD5	NULL-MD5
TLS_RSA_WITH_NULL_SHA	NULL-SHA
TLS_RSA_WITH_NULL_SHA256	NULL-SHA256
TLS_RSA_WITH_RC4_128_MD5	RC4-MD5
TLS_RSA_WITH_RC4_128_SHA	RC4-SHA
TLS_RSA_WITH_SEED_CBC_SHA	SEED-SHA

Tabulka 1.4: Části šifrovacích sad SSL/TLS podle OpenSSL. Barvy znázorňují úroveň bezpečnosti z OpenSSL 1.0.2g (vysoká, střední, nízká).

Název (OpenSSL)	Výměna klíčů	Auth.	Šifra	MAC
ADH-AES128-GCM-SHA256	DH	None	AESGCM(128)	AEAD

Tabulka 1.4: Části šifrovacích sad SSL/TLS podle OpenSSL. Barvy znázorňují úroveň bezpečnosti z OpenSSL 1.0.2g (vysoká, střední, nízká).

Název (OpenSSL)	Výměna klíčů	Auth.	Šifra	MAC
ADH-AES128-SHA	DH	None	AES(128)	SHA1
ADH-AES128-SHA256	DH	None	AES(128)	SHA256
ADH-AES256-GCM-SHA384	DH	None	AESGCM(256)	AEAD
ADH-AES256-SHA	DH	None	AES(256)	SHA1
ADH-AES256-SHA256	DH	None	AES(256)	SHA256
ADH-CAMELLIA128-SHA	DH	None	Camellia(128)	SHA1
ADH-CAMELLIA256-SHA	DH	None	Camellia(256)	SHA1
ADH-DES-CBC3-SHA	DH	None	3DES(168)	SHA1
ADH-DES-CBC3-SHA	DH	None	3DES(168)	SHA1
ADH-RC4-MD5	DH	None	RC4(128)	MD5
ADH-RC4-MD5	DH	None	RC4(128)	MD5
ADH-SEED-SHA	DH	None	SEED(128)	SHA1
AECDH-AES128-SHA	ECDH	None	AES(128)	SHA1
AECDH-AES256-SHA	ECDH	None	AES(256)	SHA1
AECDH-DES-CBC3-SHA	ECDH	None	3DES(168)	SHA1
AECDH-NULL-SHA	ECDH	None	None	SHA1
AECDH-RC4-SHA	ECDH	None	RC4(128)	SHA1
AES128-GCM-SHA256	RSA	RSA	AESGCM(128)	AEAD
AES128-SHA	RSA	RSA	AES(128)	SHA1
AES128-SHA256	RSA	RSA	AES(128)	SHA256
AES256-GCM-SHA384	RSA	RSA	AESGCM(256)	AEAD
AES256-SHA	RSA	RSA	AES(256)	SHA1
AES256-SHA256	RSA	RSA	AES(256)	SHA256
CAMELLIA128-SHA	RSA	RSA	Camellia(128)	SHA1
CAMELLIA256-SHA	RSA	RSA	Camellia(256)	SHA1
DES-CBC3-SHA	RSA	RSA	3DES(168)	SHA1
DES-CBC3-SHA	RSA	RSA	3DES(168)	SHA1
DH-DSS-AES128-GCM-SHA256	DH/DSS	DH	AESGCM(128)	AEAD
DH-DSS-AES128-SHA	DH/DSS	DH	AES(128)	SHA1
DH-DSS-AES128-SHA256	DH/DSS	DH	AES(128)	SHA256
DH-DSS-AES256-GCM-SHA384	DH/DSS	DH	AESGCM(256)	AEAD
DH-DSS-AES256-SHA	DH/DSS	DH	AES(256)	SHA1
DH-DSS-AES256-SHA256	DH/DSS	DH	AES(256)	SHA256
DH-DSS-CAMELLIA128-SHA	DH/DSS	DH	Camellia(128)	SHA1
DH-DSS-CAMELLIA256-SHA	DH/DSS	DH	Camellia(256)	SHA1
DH-DSS-DES-CBC3-SHA	DH/DSS	DH	3DES(168)	SHA1
DH-DSS-SEED-SHA	DH/DSS	DH	SEED(128)	SHA1
DHE-DSS-AES128-GCM-SHA256	DH	DSS	AESGCM(128)	AEAD
DHE-DSS-AES128-SHA	DH	DSS	AES(128)	SHA1
DHE-DSS-AES128-SHA256	DH	DSS	AES(128)	SHA256
DHE-DSS-AES256-GCM-SHA384	DH	DSS	AESGCM(256)	AEAD
DHE-DSS-AES256-SHA	DH	DSS	AES(256)	SHA1
DHE-DSS-AES256-SHA256	DH	DSS	AES(256)	SHA256
DHE-DSS-CAMELLIA128-SHA	DH	DSS	Camellia(128)	SHA1
DHE-DSS-CAMELLIA256-SHA	DH	DSS	Camellia(256)	SHA1
DHE-DSS-SEED-SHA	DH	DSS	SEED(128)	SHA1
DHE-RSA-AES128-GCM-SHA256	DH	RSA	AESGCM(128)	AEAD
DHE-RSA-AES128-SHA	DH	RSA	AES(128)	SHA1
DHE-RSA-AES128-SHA256	DH	RSA	AES(128)	SHA256
DHE-RSA-AES256-GCM-SHA384	DH	RSA	AESGCM(256)	AEAD
DHE-RSA-AES256-SHA	DH	RSA	AES(256)	SHA1
DHE-RSA-AES256-SHA256	DH	RSA	AES(256)	SHA256
DHE-RSA-CAMELLIA128-SHA	DH	RSA	Camellia(128)	SHA1
DHE-RSA-CAMELLIA256-SHA	DH	RSA	Camellia(256)	SHA1
DHE-RSA-SEED-SHA	DH	RSA	SEED(128)	SHA1

Tabulka 1.4: Části šifrovacích sad SSL/TLS podle OpenSSL. Barvy znázorňují úroveň bezpečnosti z OpenSSL 1.0.2g (vysoká, střední, nízká).

Název (OpenSSL)	Výměna klíčů	Auth.	Šifra	MAC
DH-RSA-AES128-GCM-SHA256	DH/RSA	DH	AESGCM(128)	AEAD
DH-RSA-AES128-SHA	DH/RSA	DH	AES(128)	SHA1
DH-RSA-AES128-SHA256	DH/RSA	DH	AES(128)	SHA256
DH-RSA-AES256-GCM-SHA384	DH/RSA	DH	AESGCM(256)	AEAD
DH-RSA-AES256-SHA	DH/RSA	DH	AES(256)	SHA1
DH-RSA-AES256-SHA256	DH/RSA	DH	AES(256)	SHA256
DH-RSA-CAMELLIA128-SHA	DH/RSA	DH	Camellia(128)	SHA1
DH-RSA-CAMELLIA256-SHA	DH/RSA	DH	Camellia(256)	SHA1
DH-RSA-DES-CBC3-SHA	DH/RSA	DH	3DES(168)	SHA1
DH-RSA-SEED-SHA	DH/RSA	DH	SEED(128)	SHA1
ECDH-ECDSA-AES128-GCM-SHA256	ECDH/ECDSA	ECDH	AESGCM(128)	AEAD
ECDH-ECDSA-AES128-SHA	ECDH/ECDSA	ECDH	AES(128)	SHA1
ECDH-ECDSA-AES128-SHA256	ECDH/ECDSA	ECDH	AES(128)	SHA256
ECDH-ECDSA-AES256-GCM-SHA384	ECDH/ECDSA	ECDH	AESGCM(256)	AEAD
ECDH-ECDSA-AES256-SHA	ECDH/ECDSA	ECDH	AES(256)	SHA1
ECDH-ECDSA-AES256-SHA384	ECDH/ECDSA	ECDH	AES(256)	SHA384
ECDH-ECDSA-DES-CBC3-SHA	ECDH/ECDSA	ECDH	3DES(168)	SHA1
ECDH-ECDSA-NULL-SHA	ECDH/ECDSA	ECDH	None	SHA1
ECDH-ECDSA-RC4-SHA	ECDH/ECDSA	ECDH	RC4(128)	SHA1
ECDHE-ECDSA-AES128-GCM-SHA256	ECDH	ECDSA	AESGCM(128)	AEAD
ECDHE-ECDSA-AES128-SHA	ECDH	ECDSA	AES(128)	SHA1
ECDHE-ECDSA-AES128-SHA256	ECDH	ECDSA	AES(128)	SHA256
ECDHE-ECDSA-AES256-GCM-SHA384	ECDH	ECDSA	AESGCM(256)	AEAD
ECDHE-ECDSA-AES256-SHA	ECDH	ECDSA	AES(256)	SHA1
ECDHE-ECDSA-AES256-SHA384	ECDH	ECDSA	AES(256)	SHA384
ECDHE-ECDSA-DES-CBC3-SHA	ECDH	ECDSA	3DES(168)	SHA1
ECDHE-ECDSA-NULL-SHA	ECDH	ECDSA	None	SHA1
ECDHE-ECDSA-RC4-SHA	ECDH	ECDSA	RC4(128)	SHA1
ECDHE-RSA-AES128-GCM-SHA256	ECDH	RSA	AESGCM(128)	AEAD
ECDHE-RSA-AES128-SHA	ECDH	RSA	AES(128)	SHA1
ECDHE-RSA-AES128-SHA256	ECDH	RSA	AES(128)	SHA256
ECDHE-RSA-AES256-GCM-SHA384	ECDH	RSA	AESGCM(256)	AEAD
ECDHE-RSA-AES256-SHA	ECDH	RSA	AES(256)	SHA1
ECDHE-RSA-AES256-SHA384	ECDH	RSA	AES(256)	SHA384
ECDHE-RSA-DES-CBC3-SHA	ECDH	RSA	3DES(168)	SHA1
ECDHE-RSA-NULL-SHA	ECDH	RSA	None	SHA1
ECDHE-RSA-RC4-SHA	ECDH	RSA	RC4(128)	SHA1
ECDH-RSA-AES128-GCM-SHA256	ECDH/RSA	ECDH	AESGCM(128)	AEAD
ECDH-RSA-AES128-SHA	ECDH/RSA	ECDH	AES(128)	SHA1
ECDH-RSA-AES128-SHA256	ECDH/RSA	ECDH	AES(128)	SHA256
ECDH-RSA-AES256-GCM-SHA384	ECDH/RSA	ECDH	AESGCM(256)	AEAD
ECDH-RSA-AES256-SHA	ECDH/RSA	ECDH	AES(256)	SHA1
ECDH-RSA-AES256-SHA384	ECDH/RSA	ECDH	AES(256)	SHA384
ECDH-RSA-DES-CBC3-SHA	ECDH/RSA	ECDH	3DES(168)	SHA1
ECDH-RSA-NULL-SHA	ECDH/RSA	ECDH	None	SHA1
ECDH-RSA-RC4-SHA	ECDH/RSA	ECDH	RC4(128)	SHA1
EDH-DSS-DES-CBC3-SHA	DH	DSS	3DES(168)	SHA1
EDH-DSS-DES-CBC3-SHA	DH	DSS	3DES(168)	SHA1
EDH-RSA-DES-CBC3-SHA	DH	RSA	3DES(168)	SHA1
EDH-RSA-DES-CBC3-SHA	DH	RSA	3DES(168)	SHA1
NULL-MD5	RSA	RSA	None	MD5
NULL-MD5	RSA	RSA	None	MD5
NULL-SHA	RSA	RSA	None	SHA1
NULL-SHA	RSA	RSA	None	SHA1
NULL-SHA256	RSA	RSA	None	SHA256

Tabulka 1.4: Části šifrovacích sad SSL/TLS podle OpenSSL. Barvy znázorňují úroveň bezpečnosti z OpenSSL 1.0.2g (vysoká, střední, nízká).

Název (OpenSSL)	Výměna klíčů	Auth.	Šifra	MAC
PSK-3DES-EDE-CBC-SHA	PSK	PSK	3DES(168)	SHA1
PSK-AES128-CBC-SHA	PSK	PSK	AES(128)	SHA1
PSK-AES256-CBC-SHA	PSK	PSK	AES(256)	SHA1
PSK-RC4-SHA	PSK	PSK	RC4(128)	SHA1
RC4-MD5	RSA	RSA	RC4(128)	MD5
RC4-MD5	RSA	RSA	RC4(128)	MD5
RC4-MD5	RSA	RSA	RC4(128)	MD5
RC4-SHA	RSA	RSA	RC4(128)	SHA1
RC4-SHA	RSA	RSA	RC4(128)	SHA1
SEED-SHA	RSA	RSA	SEED(128)	SHA1

1.3 SSH

SSH (Secure Shell) zprostředkovává zabezpečený kanál mezi klientem a serverem [44]. Standardní využití je vzdálená příkazová řádka.

Protokol SSH lze ale použít i na [44]:

- tunelování,
- přesměrování TCP portů,
- X11 spojení (X11 je okénkový systém - lze tedy přenášet i okenní aplikace),
- přenášení souborů (SCP[17], SFTP[18]).

1.3.1 Verze SSH

SSH verze 1.0

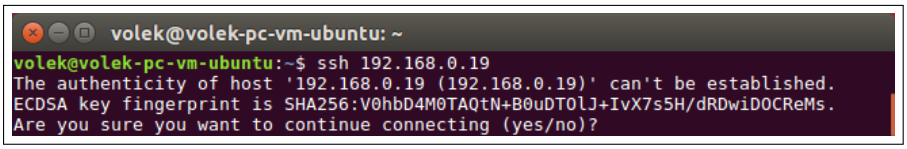
SSH verze 1 vznikla v roce 1995 za účelem nahrazení nezabezpečených vzdálených příkazových řádků (Telnet, rlogin, rsh).

Od roku 2001 je známa zranitelnost protokolu SSH verze 1 vůči útoku MitM [45, 46]. Z tohoto důvodu je doporučené používat SSH verze 2 a verzi SSH 1 úplně zakázat.

SSH verze 2

SSH protokol verze 2 je nástupce verze 1. Verze 2 není kompatibilní s verzí 1. SSH 2 je definována standardem RFC a má následující interní architekturu.

[17] Secure copy
[18] Secure File Transfer Protocol



Obrázek 1.2: Ukázka potvrzení fingerprintu při připojení na SSH server

Interní architektura SSH 2:

- přenosová (transportní) vrstva - RFC 4253 [47],
- ověření uživatele (autentizace) - RFC 4252 [44],
- spojová (connection) vrstva - RFC 4254 [48].

1.3.2 Připojení na SSH

Při prvním připojování na SSH server musí klient potvrdit otisk veřejného klíče (key fingerprint), viz obrázek 1.2. V případě útoku MitM je ale veřejný klíč podvržený, má tak jiný otisk. Je ale běžným zvykem, že to lidé příliš nekontrolují. Případně si při prvním spojení zkontrolují jen pár prvních a posledních znaků otisku.

Je ale možné ověřovat fingerprint pomocí SSHFP[19] DNS[20] záznamu (RFC 4255 [49]). To znamená, že se vytvoří pro danou adresu tento záznam v DNS tabulce a ověření proběhne automatiky. Je zde ale nutná podpora SSH klienta. Tuto metodu je vhodné kombinovat s DNSSEC[21], který zajišťuje integritu DNS záznamů.

Ukázka SSHFP DNS záznamu

```
host.example.com.  SSHFP 2 1 123456789abcdef67890123456789ab...
```

1.4 VPN protokoly

Existuje řada protokolů pro VPN (Virtual Private Network). VPN umožňuje lidem připojovat se vzdáleně do například firemní sítě, když jsou mimo kancelář. V tomto případě je téměř nutností zajistit bezpečnost a důvěryhodnost takového připojení [50, 51]. V této práci se budu věnovat těmto VPN službám: PPTP, OpenVPN a L2TP/IPsec.

[19] SSH Fingerprint
[20] Domain Name System
[21] Domain Name System Security Extensions

1.4.1 PPTP

PPTP (Point-to-Point Tunneling Protocol) je zastaralý VPN protokol definovaný v RFC 2637 [52]. PPTP je nativně podporovaný ve Windows od verze 95. Je velmi jednoduchý na konfiguraci a používání.

Bezpečnost PPTP

V roce 2012 byl prolomen protokol MS-CHAPv2[22] [53]. Od této doby již není protokol PPTP považován za bezpečný a tedy na komunikaci přes PPTP lze nahlížet jako na nešifrovanou. Místo protokolu PPTP by měl být použit jiný VPN protokol, např. OpenVPN nebo L2TP[23]/IPSec.

1.4.2 OpenVPN

OpenVPN je open source technologie k vytváření VPN spojení. OpenVPN umožňuje vytvářet VPN tunely na vrstvě 2 (Ethernet tunel) i na vrstvě 3 (IP tunel) pětivrstvého síťového modelu.

OpenSSL může běžet na libovolném portu a podporuje protokoly TCP i UDP[24]. Je tedy možné vést VPN spojení přes NAT[25] a firewall, protože se komunikace jeví třeba jako normální webová komunikace na portu 443 (standardně HTTPS).

OpenVPN je rychlý VPN protokol, je rychlejší než PPTP i L2TP [54]. Na rozdíl od VPN typů PPTP a L2TP/IPsec není OpenVPN nativně podporován v operačním systému Windows. Je tedy nutné instalovat speciálního klienta.

Bezpečnost OpenVPN

OpenVPN je postavena na protokolech SSL/TLS a používá knihovnu OpenSSL [55]. Bezpečnost OpenVPN vychází z bezpečnosti protokolů SSL/TLS a knihovny OpenSSL. Ohledně bezpečnosti OpenVPN tedy platí vše, co bylo diskutováno v sekci 1.2 věnované protokolům SSL/TLS.

1.4.3 L2TP/IPsec

Protokol L2TP (Layer 2 Tunneling Protocol) je VPN protokol, který pracuje na 3. (síťové) vrstvě pětivrstvého síťového modelu. To neodpovídá názvu (Layer 2 ...), protože název pracuje s čtyřvrstvým modelem (TCP/IP model). L2TP samo o sobě nepodporuje žádné šifrování, protokol umožňuje jen tunelování [56]. Z tohoto důvodu se právě používá společně s jiným protokolem

[22] Microsoft Challenge-Handshake Authentication Protocol
[23] Layer 2 Tunneling Protocol
[24] User Datagram Protocol
[25] Network address translation

zajišťujícím šifrování a bezpečnost. L2TP se používá společně se skupinou protokolů IPsec.

Vzhledem k tomu, že pakety musí projít dvěma procesy - zabalení do L2TP a pak do IPsec, tak L2TP je o něco pomalejší, než například OpenVPN [54].

IPsec

IPsec je skupina protokolů, zajišťujících zabezpečení IP komunikace. Jedná se tedy o rozšíření IP protokolu o autentizaci a šifrování na 3. (síťové) vrstvě pětivrstvého síťového modelu. IPsec lze použít jako rozšíření pro IPv4. V případě nové verze IP protokolu je IPsec přímo součástí specifikace IPv6 [57]. IPsec je definovaný v desítkách RFC dokumentů, z nichž hlavní jsou RFC 2401 [58] a RFC 2411 [59].

V rámci handshaku a následné komunikace lze použít velkou řadu algoritmů pro výměnu klíčů, autorizaci a šifrování. Tyto podporované algoritmy jsou definovány v RFC 7321 [60]. U každého algoritmu je zde definováno, zda musí/může/nesmí být podporován. Je tedy nutné správně konfigurovat provozovanou VPN službu tak, aby byly povolené jen bezpečné algoritmy.

Módy IPsec

IPsec může pracovat ve dvou módech [61]:

- **Tunnel mode** - celý původní paket je zašifrován a je přidána nová hlavička. Výhoda tohoto módu je ta, že nelze odhalit IP adresu odesílatele.
- **Transport mode** - šifrují se pouze data. IP hlavička zůstává a doplní se IPsec hlavička. Zde pak pak odhalitelná IP adresa odesílatele. Tento mód je využíván v nativním L2TP/IPsec klientu ve Windows.

Protokoly IPsec

Jak již bylo zmíněno výše, tak IPsec je skupina protokolů [61]:

- **Authentication Header** (AH) - dnes už je zastaralý a příliš se nepoužívá. Zajišťuje integritu a autentizaci zdroje. Používá na začátku domluvený společný klíč hashovací funkce. Do hlavičky přidává pořadové číslo paketu. Využívá protokol IP číslo 51.
- **Encapsulating Security Payload** (ESP) - Zajišťuje integritu a autentizaci zdroje. Používá šifrovací algoritmy. Využívá protokol IP 50.

Security Association

Security Association (SA) je skupina algoritmů a parametrů (např. klíčů), které IPsec používá k autentizaci a šifrování komunikace. Na SA se jednotlivé strany domluví pomocí protokolu ISAKMP[26]/IKE[27] [62]. SA je využívána jen pro komunikaci jedním směrem. Obousměrná komunikace je tedy zabezpečena pomocí dvojice SA.

[26]Internet Security Association and Key Management Protocol
[27]Internet Key Exchange

Kapitola 2

Návrh testovacího nástroje

Cílem této práce je vytvořit nástroj pro automatizované testování bezpečného nastavení vybraných služeb se šifrovanými protokoly. Existuje již mnoho nástrojů pro jednotlivé testování bezpečné konfigurace pro často používané protokoly (například skripty programu nmap [63]). Účelem tohoto nového nástroje je automatizace těchto testů a zjednodušené vyhodnocování, zda nastavení jsou bezpečná.

Využití nástroje

Tento nástroje funguje tak, že je administrátorem nakonfigurován, aby pravidelně testoval bezpečnou konfiguraci šifrovaných služeb. Nástroj tedy lze použít jak pro prvotní otestování správné konfigurace, tak i následně dokáže odhalit nevhodnou změnu konfigurace způsobenou například neodborným zásahem, aktualizací softwaru nebo jinou těžko předpovídatelnou změnou.

Když nastane nějaká změna a nástroj vyhodnotí, že změna má charakter bezpečnostní hrozby, upozorní správce emailovou notifikací, která obsahuje podrobné výsledky testu, který byl takto vyhodnocen.

Typický příklad využití tohoto nástroje je následující: Správce, který zprovoznil nějakou novou šifrovanou službu, přidá do konfigurace zde vyvinutého nástroje test nebo testy, kterými bude služba pravidelně testována. Správce je pak automaticky upozorněn, pokud nastane změna, která může mít charakter bezpečnostní hrozby.

Jméno nástroje

Vzhledem k tomu, že se jedná o nástroj, který testuje protokoly, rozhodl jsem se ho pojmenovat **Prottest**. Prottest představuje zkratku pro *protocol test*. Původně jsem zvažoval název jen s jedním „t“ - Protest. To jsem nakonec

zavrhl, protože již existuje balíček s tímto názvem v balíčkovacím systému `npm` [64], což by mohlo uživatele mást.

2.1 Volba technologií

Cílovou platformou, na které bude nástroj běžet, je Linux. Proto jsem vybíral z následujících technologií.

Bash

Prvním nápadem bylo použít Bash (Bourne Again Shell) [65], to jsem ale zavrhl, protože Bash mi přijde vhodný spíše na jednorázové skripty, než na komplexnější rozšiřitelný program. Mým záměrem bylo vytvořit program se složitější rozšiřitelnou architekturou a na to by byl Bash nevhodný.

Python

Další možnou volbou byl Python [66]. Python už by umožňoval vytvořit rozsáhlejší rozšiřitelnou architekturu. U Pythonu mi ale nevyhovuje dynamická typová kontrola až při běhu. Jako vývojáři mi mnohem více vyhovuje statická typová kontrola při kompilaci, což zvolené řešení (NodeJS + TypeScript) alespoň částečně umožňuje.

.NET Core

.NET Core [67] je nová technologie umožňující používat C# a .NET framework na Linuxu. Tuto technologii jsem hodně zvažoval. C# a .NET jsou mojí nejoblíbenější technologií, protože jsou dle mých zkušeností pro vývojáře nejpohodlnější. .NET Core pro Linux je ale bohužel ještě v nefinální verzi a není příliš rozšířen. Dále jsem nechtěl podmiňovat použití nástroje instalací .NET Core runtimu, proto jsem jej nakonec nezvolil.

NodeJS + TypeScript

NodeJS [68] je JavaScriptový runtime, který se často používá pro psaní webových serverů. Je ale možné pomocí něj psát i konzolové a okenní aplikace [69]. JavaScript je skriptovací jazyk bez statické typové kontroly při kompilaci.

Nadstavba JavaScriptu jménem TypeScript [70] od firmy Microsoft v jisté míře zavádí statickou typovou kontrolu a rozšiřuje JavaScript o další syntaktické prostředky a tím odstraňuje některé nedostatky JavaScriptu.

Zvolená technologie

Volil jsem tedy mezi technologiemi Bash, Python, .NET Core a NodeJS + TypeScript. Nakonec jsem zvolil kombinaci NodeJS + TypeScript. Měl jsem

s touto technologií již nějaké zkušenosti a bylo to z výše uvedených důvodů nejvhodnější pro tuto práci.

2.2 Testování nastavení zabezpečených protokolů

Primárním účelem tohoto nástroje je testování správné konfigurace bezpečnosti šifrovaných protokolů. V implementaci bude tedy každý konkrétní test reprezentovaný konkrétní implementací nějakého rozhraní, případně bázové třídy, tzv. testovacím modulem.

2.2.1 Testovací moduly

Pro každý jednotlivý test tedy existuje testovací modul, který má jednotné programové rozhraní (API[28]). Jednotné API spočívá v abstraktní metodě, která spouští konkrétní test a vrací jeho výsledek.

Testovací modul provede testování a vrátí výsledek. O další zpracování výsledku se již nestará. Zpracování výsledku je obstaráno jinou částí aplikace. Objektová struktura výsledku všech testovacích modulu je stejná, aby bylo možné výsledky zpracovávat stejným způsobem.

Výsledky testů

Pro výsledky testovacích modulů je navržena jednotná objektová struktura. Jedná se o strom, protože výsledky některých složitějších testů mohou mít rozsáhlou strukturu (například vylistování podporovaných šifer SSL/TLS spolu s jejich sílou).

Naopak jednoduché testy mohou mít ploché výsledky typu ano/ne (například SSH verze 1 ano/ne). Je tedy na testovacím modulu, zda výsledkem bude jen jeden uzel, nebo složitá stromová struktura.

Každý uzel výsledku je složen z textové zprávy a typu zprávy. Pokud má uzel potomky, je typ zprávy určen dle nejhoršího typu zprávy svých potomků. Potomky jsou v uzlu reprezentovány pole, každý uzel tedy může mít různý počet potomků.

Typy zprávy výsledku

- **info** - informační zpráva,
- **ok** - přijatelný výsledek,
- **warning** - varování,
- **critical** - kritická bezpečnostní chyba,
- **fail** - test nedoběhl z důvodu neočekávané chyby.

[28] Application programming interface

Ohodnocení síly bezpečnostních algoritmů

V některých testech se ohodnocují síly algoritmů pro výměnu klíčů, autentizaci, šifrování a MAC, případně síly celé šifrovací sady.

Stupnice síly algoritmů

- **high** - vysoká úroveň bezpečnosti,
- **medium** - střední úroveň bezpečnosti,
- **low** - nízká úroveň bezpečnosti.

Tato stupnice je převzatá z úrovní OpenSSL (viz 1.2.4). OpenSSL ohodnocuje pomocí této stupnice jen s tím rozdílem, že OpenSSL má ještě úroveň null - to znamená žádná úroveň bezpečnosti (například žádné šifrování - nešifrovaný přenos). Tato null úroveň je zde shrnuta pod úroveň low.

Mapování síly algoritmu na typ zprávy výsledku

- **high** - **ok**,
- **medium** - **warning**,
- **low** - **critical**.

Příklad jednoduché stromové struktury výsledku V tomto příkladu se jedná výsledek testu, který ověřuje zda SSH server podporuje SSH verzi 1.

```
Sshv1Tester (localhost:22)
    SSHv1 not supported.
```

Příklad složitější stromové struktury výsledku V tomto příkladu se jedná výsledek testu, který ohodnocuje šifrovací sady podporované konkrétním SSL/TLS serverem. V tomto testy jsou napsány názvy testovacích sad tak, jak jsou definovány v RFC.

```
SslTlsCipherTester (localhost:443)
    Evaluator: Custom
    Definition file: SslCustomCipherSuitesStrengts.xml
    TLSv1.0
        Cipher preference: server
        Ciphers:
            TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA : high
            TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA : high
            TLS_DHE_RSA_WITH_AES_256_CBC_SHA : high
```

```
            TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA : high
            TLS_DHE_RSA_WITH_AES_128_CBC_SHA : high
            TLS_DHE_RSA_WITH_SEED_CBC_SHA : low
            TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA : high
    TLSv1.1
        Cipher preference: server
        Ciphers:
            TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA : high
            TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA : high
            TLS_DHE_RSA_WITH_AES_256_CBC_SHA : high
            TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA : high
            TLS_DHE_RSA_WITH_AES_128_CBC_SHA : high
            TLS_DHE_RSA_WITH_SEED_CBC_SHA : low
            TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA : high
```

Zpracování výsledků

Výsledky všech testovacích modulů mají stejnou objektovou stromovou strukturu. To umožňuje jednotně zpracovávat výsledky různých testovacích modulů.

V případě, že jsou testy spouštěny jednorázově, tak se výsledky vypisují na standardní výstup.

V případě, že je nástroj spuštěn jako bezobslužná služba a testování je prováděno automatiky, jsou výsledky ukládány do souboru. Takto uložené výsledky jsou pak zpřístupněny pomocí webového rozhraní.

2.2.2 Spouštění testovacích modulů

Testovací moduly lze spouštět třemi způsoby:

- jednorázové spuštění konkrétního testovacího modulu z příkazové řádky,
- jednorázové spuštění testovacích modulů definovaných v konfiguračním souboru,
- spuštění služby, které bude opakovaně spouštět testovací moduly definované v konfiguračním souboru.

Jednorázové spuštění konkrétního testu

Základním způsobem spuštění testu je spuštění přímo z příkazové řádky. Je zde nutné specifikovat název testu a také všechny potřebné parametry pro test. Konfigurační soubor je v tomto případě ignorován. Výstup je v tomto případě vypsán přímo na standardní výstup.

Jednorázové spuštění testů z konfigurace

Dále lze spustit sadu testů definovanou v konfiguračním souboru. Ostatní části konfiguračního souboru kromě sady testů jsou v tomto případě ignorovány. Výstup je i zde vypsán přímo na standardní výstup.

Opakované spuštění testů z konfigurace

Hlavním způsobem použití tohoto nástroje je spuštění jako služby, která bude v pravidelných intervalech spouštět sadu testů z konfiguračního souboru.

Výsledky nyní nejsou vypisovány na standardní výstup, ale jsou ukládány do permanentního úložiště (do souboru). Služba tyto výsledky zpřístupňuje pomocí webového serveru, který je spuštěn v době běhu služby.

V případě, že se objeví nějaké bezpečností riziko, je administrátor upozorněn emailem s konkrétním problematickým výsledkem testu.

2.3 Navržené testovací moduly

2.3.1 Testovací modul `SslTlsCipherTester`

Testovací modul `SslTlsCipherTester` testuje podporované verze protokolů SSL/TLS, jejich podporované šifrovací sady (cipher suites). U jednotlivých šifrovacích sad ohodnocuje jejich sílu (bezpečnost).

Dále tento testovací modul ověřuje, zda výběr šifrovací sady je na serveru nebo na klientovi.

Tento tester využívá skriptu `ssl-enum-ciphers` programu nmap. Výstupem tohoto skriptu je seznam šifrovacích sad pro jednotlivé verze protokolů. Tento výstup je zpracován a dále jsou jednotlivé šifrovací sady ohodnoceny.

Tento tester umožňuje ohodnotit šifrovací sady dvěma způsoby v závislosti na vstupních parametrech.

Možnosti ohodnocení šifrovací sady

- OpenSSL,
- vlastní soubor definující bezpečné prvky šifrovací sady.

Ohodnocení dle OpenSSL

Knihovna OpenSSL obsahuje seznamy šifrovacích sad ve třech různých úrovních bezpečnosti. Jaké šifrovací sady mají dle OpenSSL jakou úroveň bezpečnosti je barevně znázorněno v analýze tabulce 1.3.

Zde nastává ale problém, že název šifrovacích sad je v OpenSSL rozdílný od názvů z RFC, které používá právě nmap. Je tedy nutné překládat z OpenSSL názvů na RFC názvy.

Toto je vyřešeno tak, že součástí programu Prottest je textový soubor s překladovým slovníkem názvů šifrovacích sad z OpenSSL názvů do RFC názvů. Překladový soubor obsahuje dva sloupce oddělené dvojtečkou. V prvním sloupci je OpenSSL název a v druhém sloupci je RFC název šifrovací sady. Při ohodnocování šifry se tedy napřed v tomto slovníku najde k RFC názvu OpenSSL název a ten se poté ohodnocuje dle OpenSSL. Příklad slovníkového souboru je v ukázce souboru 1. Kompletní slovník je v analýze v tabulce 1.3.

Ukázka souboru 1: Překladový soubor z RFC do OpenSSL názvů

```
AES128-SHA:TLS_RSA_WITH_AES_128_CBC_SHA
AES256-SHA:TLS_RSA_WITH_AES_256_CBC_SHA
DH-DSS-AES128-SHA:TLS_DH_DSS_WITH_AES_128_CBC_SHA
DH-DSS-AES256-SHA:TLS_DH_DSS_WITH_AES_256_CBC_SHA
DH-RSA-AES128-SHA:TLS_DH_RSA_WITH_AES_128_CBC_SHA
DH-RSA-AES256-SHA:TLS_DH_RSA_WITH_AES_256_CBC_SHA
...
```

Ohodnocení dle vlastního souboru

Dále je možné ohodnocovat šifry dle vlastního definičního souboru.

Účelem možnosti definovat vlastní hodnocení síly šifrovacích sad je ten, aby mohl uživatel/správce kontrolovat správné nastavení za základě usnesení v zákoně o kybernetické bezpečnosti [27, 28] nebo například dle interních bezpečnostních pravidel v organizaci.

Pokud by ale v tomto konfiguračním souboru byly vypsány celé názvy šifrovacích sad, tak by konfigurace byla velmi nepohodlná a složitá. Šifrovacích sad je totiž hodně. Dále se šifrovací sada se skládá z několika částí, které se často opakují. Je tedy pro uživatele pohodlnější a přehlednější určovat úroveň bezpečnosti pro jednotlivé části zvlášť.

Aby tedy mohla být vyhodnocena úroveň bezpečnosti celé šifrovací sady, je nutné ji „roztrhnout“ na jednotlivé části. Samotné roztrhnutí není provedeno parsováním RFC nebo OpenSSL názvu, ale je získáno opět pomocí knihovny OpenSSL. OpenSSL umožňuje pomocí příkazu „`openssl ciphers -v`“ vypsat k OpenSSL názvu jednotlivé části šifrovací sady [43]. Tento výstup je zpracován a tím jsou získány jednotlivé části šifrovací sady, které jsou poté srovnány s definičním souborem. Mapování jednotlivých šifrovacích sad na konkrétní části je v analýze v tabulce 1.4.

V tomto definičním souboru se tedy nedefinuje úroveň bezpečnosti celých šifrovacích sad, ale konkrétně jednotlivých částí. Příklad konfiguračního souboru je v ukázce souboru 2 na straně 35.

V definičním souboru se ohodnocují algoritmy dle stupnice high, medium a low popsanou výše. Definují se zde pouze stupně high a medium. Všechny

ostatní algoritmy neobsažené v sekcích high a medium jsou automaticky považovány za stupeň low.

Jednotlivé části šifrovací sady

- algoritmus pro výměnu klíčů (`keyExchange`),
- algoritmus pro podpis (`authentication`),
- šifrovací algoritmus (`keyExchange`),
- MAC (`messageAuthentication`).

2.3.2 Testovací modul `SslTlsCertTester`

Testovací modul `SslTlsCertTester` kontroluje certifikát SSL/TLS. V první řadě kontroluje, zda je certifikát platný časově.

Dále v případě, že je testovacímu modulu parametrem předán očekávaný SHA1 hash serverového certifikátu, tester zkontroluje, zda zadaný hash odpovídá hashi certifikátu na serveru. Tímto způsobem by byla odhalena neoprávněná změna certifikátu nebo například útok MitM.

2.3.3 Testovací modul `SshCipherTester`

Testovací modul `SshCipherTester` kontroluje a ohodnocuje nastavení algoritmů pro výměnu klíčů, autentizaci, šifrování a MAC protokolu SSH.

Jednotlivé části šifrovacích sad je zde možné ohodnocovat dle definičního souboru. Názvy jsou z RFC 4253 [47]. Příklad takového souboru je v ukázce souboru 3 na straně 36.

V definičním souboru se ohodnocují algoritmy dle stupnice high, medium, low pospanou výše. Definují se zde pouze stupně high a medium. Všechny ostatní algoritmy jsou automaticky považovány za stupeň low.

2.3.4 Testovací modul `Sshv1Tester`

Testovací modul `Sshv1Tester` kontroluje, zda není na SSH serveru povolena starší verze SSHv1, která již není považována za bezpečnou [45, 46]. Pokud je povolena SSHv1, je to vyhodnoceno jako kritická bezpečností hrozba.

2.3.5 Testovací modul `SshHostKeyTester`

Testovací modul `SshHostKeyTester` přijímá parametrem očekávané otisky klíčů (host key fingerprints) SSH serveru a kontroluje, zda nedošlo ke jejich změně. Tímto způsobem by byla odhalena neoprávněná změna klíčů nebo například útok MitM.

Ukázka souboru 2: Definiční soubor pro `SslTlsCipherTester`

```
<?xml version="1.0" encoding="UTF-8"?>
<cipherSuitesStrength>
    <keyExchange>
        <high>
            <item>DH</item>
            <item>ECDH</item>
            ...
        </high>
        <medium></medium>
    </keyExchange>

    <authentication>
        <high>
            <item>ECDSA</item>
            <item>RSA</item>
            ...
        </high>
        <medium></medium>
    </authentication>

    <encryption>
        <high>
            <item>AES(128)</item>
            <item>AES(192)</item>
            ...
        </high>
        <medium>
            <item>3DES(168)</item>
        </medium>
    </encryption>

    <messageAuthentication>
        <high>
            <item>SHA2</item>
            <item>SHA256</item>
            ...
        </high>
        <medium>
            <item>SHA1</item>
        </medium>
    </messageAuthentication>
</cipherSuitesStrength>
```

Ukázka souboru 3: Definiční soubor pro `SshCipherTester`

```
<?xml version="1.0" encoding="UTF-8"?>
<cipherSuitesStrength>
    <keyExchange>
        <high>
            <item>ecdh-sha2-nistp256</item>
            <item>ecdh-sha2-nistp384</item>
            ...
        </high>
        <medium></medium>
    </keyExchange>

    <authentication>
        <high>
            <item>ssh-rsa</item>
            <item>rsa-sha2-512</item>
            ...
        </high>
        <medium>
            <item>ssh-ed25519</item>
        </medium>
    </authentication>

    <encryption>
        <high>
            <item>aes128-ctr</item>
            <item>aes192-ctr</item>
            ...
        </high>
        <medium></medium>
    </encryption>

    <messageAuthentication>
        <high>
            <item>*</item>
        </high>
        <medium></medium>
    </messageAuthentication>
</cipherSuitesStrength>
```

Kapitola 3

Implementace testovacího nástroje

3.1 Vývojové prostředí

Jak bylo zmíněno v kapitole 2 s návrhem, k implementaci bude využita technologie NodeJS a TypeScript. Pro tuto kombinaci jsem využil poměrně nové vývojové prostředí Visual Studio Code od společnosti Microsoft [71].

Překlad TypeScriptu do JavaScriptu

Překlad TypeScriptu do JavaScriptu je proveden pomocí kompilátoru `tsc`. Celá kompilace je prováděna pomocí nástroje gulp.js. Gulp.js je nástroj umožňující automatizaci opakovaných akcí při kompilaci [72].

Kontrola konvencí TypeScriptu

TypeScript sám o sobě podporuje statickou typovou kontrolu při kompilaci. Při vývoji tohoto nástroje byl využit tzv. linting. Linting je statická kontrola dodržování konvencí kódu nad rámec standardní syntaktické kontroly. To přispívá k čistotě kódu a pomáhá předcházet chybám.

3.2 Testovací moduly

Každý testovací modul je reprezentován potomkem třídy Tester. Bázová třída tester obsahuje:

- metoda `run()`,
- abstraktní metoda `runTest()`,
- metoda `requiredCommands()`.

Metoda `runTest()`

Metoda `runTest()` je v bázové třídě abstraktní a je nutné ji tedy implementovat ve všech konkrétních testerech. Implementace v konkrétních testovacích modulech provádí skutečné testování a vrací výsledky tohoto testování.

Testování v této metodě může probíhat jakýmkoli způsobem. Standardní implementace, která byla využita pro implementaci testovacích modulů v této práci, vypadá tak, že se spustí nějaký příkaz v bashi jako další proces. Poté se zpracuje výstup tohoto procesu a transformuje se do jednotné struktury výsledků (viz sekce 2.2.1).

Metoda `requiredCommands()`

Tato metoda vrací pole příkazů, které jsou nutné pro běh testu (např. `openssl` nebo `nmap`). V bázové třídě vrací standardně prázdné pole. V případě potřeby se v potomcích přetíží, aby vracela skutečný seznam příkazů, který test vyžaduje. Bázová třída se sama postará, aby před spuštěním testu (metody `runTest()`) bylo automaticky ověřeno, že příkaz je na daném stroji dostupný. Pokud příkaz není dostupný, test automaticky skončí s výsledkem `fail`.

Metoda `run()`

Touto veřejnou metodou se spouští konkrétní test. Tato metoda jen zabaluje metodu `runTest()` a zajišťuje okolnosti nutné pro běh testu. Například zpracovává výjimky a ověřuje, zda jsou k dispozici všechny příkazy, které vrací metoda `requiredCommands()`.

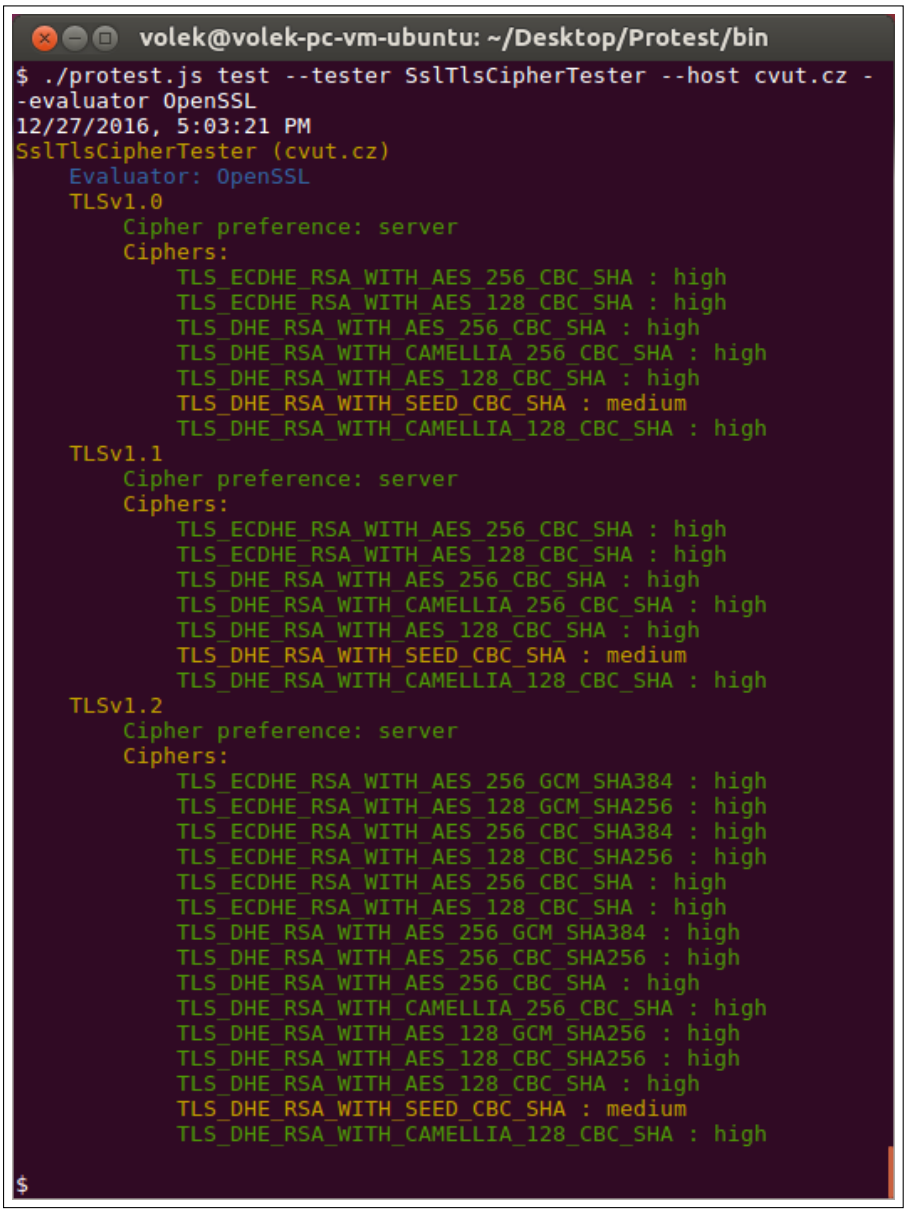
3.3 Implementované testovací moduly

3.3.1 `SslTlsCipherTester`

Testovací modul `SslTlsCipherTester` testuje podporované verze protokolů SSL/TLS a jejich podporované šifrovací sady (cipher suites). U jednotlivých šifrovacích sad ohodnocuje jejich sílu (bezpečnost). Ukázka běhu tohoto testu je na obrázku 3.1.

Návrh tohoto testovacího modulu je v sekci 2.3.1. Test je implementován pomocí následujících kroků:

1. Získání nastavení SSL/TLS serveru, nastavení je získano pomocí skriptu `ssl-enum-ciphers` programu `nmap`,
2. ohodnocení jednotlivých šifrovacích sad dle vstupních parametrů testovacího modulu:
   - ohodnocení dle OpenSSL,
   - ohodnocení dle vlastního souboru.



Obrázek 3.1: Běh testu `SslTlsCipherTester` v příkazové řádce

Získání nastavení SSL/TLS serveru

V prvním kroku testu jsou zjištěny povolené šifrovací sady SSL/TLS serveru pomocí skriptu `ssl-enum-ciphers` [73] programu `nmap` [74]. Výstup tohoto skriptu je rozparsován do objektové formy vhodné pro další zpracování.

Program `nmap` umožňuje vypsat na standardní výstup jen klasický textový formát. Tento formát není pro parsování příliš vhodný. Program `nmap` umožňuje i XML[29] výstup do souboru. Tento XML výstup je pro parsování vhodnější, je tedy použito tohoto výstupu. V průběhu testu je tedy vytvořen dočasný XML soubor, který je po dokončení testu smazán.

Výstupem tohoto kroku je objektová struktura, která obsahuje získané informace ze spuštěného příkazu. Tato objektová struktura obsahuje mimo jiné právě jednotlivé šifrovací sady, které budou v následujícím kroku ohodnoceny.

Ohodnocení šifrovací sady

Rozhraní `ICipherSuiteStrengthEvaluator` obsahuje jednu metodu, která přijímá RFC název šifrovací sady jako parametr a vrací sílu šifry dle stupnice uvedené v sekci 2.2.1 na straně 30. Každý typ ohodnocení (OpenSSL, soubor) implementuje toto rozhraní. Způsob ohodnocování šifrovací sady je tedy navenek jednotný, protože obě implementace mají stejné rozhraní (API).

Ohodnocení dle OpenSSL

První možností je ohodnocení jednotlivých šifrovacích sad dle OpenSSL. Jak je zmíněno v analýze v sekci 1.2.4.1, tak OpenSSL umožňuje ohodnotit jednotlivé šifry pomocí stupnice high, medium a low.

Zde se konkrétní implementace rozhraní `ICipherSuiteStrengthEvaluator` jmenuje `OpenSslCipherSuiteEvaluator`. Ohodnocení šifrovací sady zde funguje ve dvou krocích:

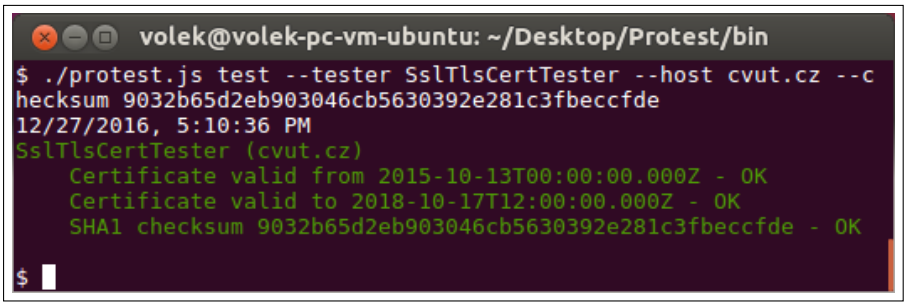
1. překlad z RFC názvu šifrovaí sady do OpenSSL názvu (viz sekce 1.2.4.2),
2. ohodnocení šifrovací sady dle OpenSSL (viz sekce 1.2.4.1).

Ohodnocení dle vlastního souboru

Druhou možností je ohodnotit šifrovací sadu dle vlastního konfiguračního souboru. Jak je uvedeno v návrhu, tak v konfiguračním souboru nejsou ohodnoceny celé šifrovací sady, ale jednotlivě jejich jednotlivé části (algoritmy pro výměnu klíčů, autentizaci, šifrování a MAC).

Konkrétní implementace rozhraní `ICipherSuiteStrengthEvaluator` se zde jmenuje `CustomCipherSuiteEvaluator`. Ohodnocení šifrovací sady zde funguje ve čtyřech krocích:

[29] Extensible Markup Language



Obrázek 3.2: Běh testu `SslTlsCertTester` v příkazové řádce

1. překlad z RFC názvu šifrování sady do OpenSSL názvu,
2. rozpad OpenSSL názvu do jednotlivých částí šifrovací sady pomocí OpenSSL,
3. načtení definičního souboru,
4. ohodnocení jednotlivých částí šifrovací sady dle definičního souboru. Výsledné ohodnocení šifrovací sady je určeno jako nejnižší hodnota z ohodnocení všech jejích částí.

3.3.2 `SslTlsCertTester`

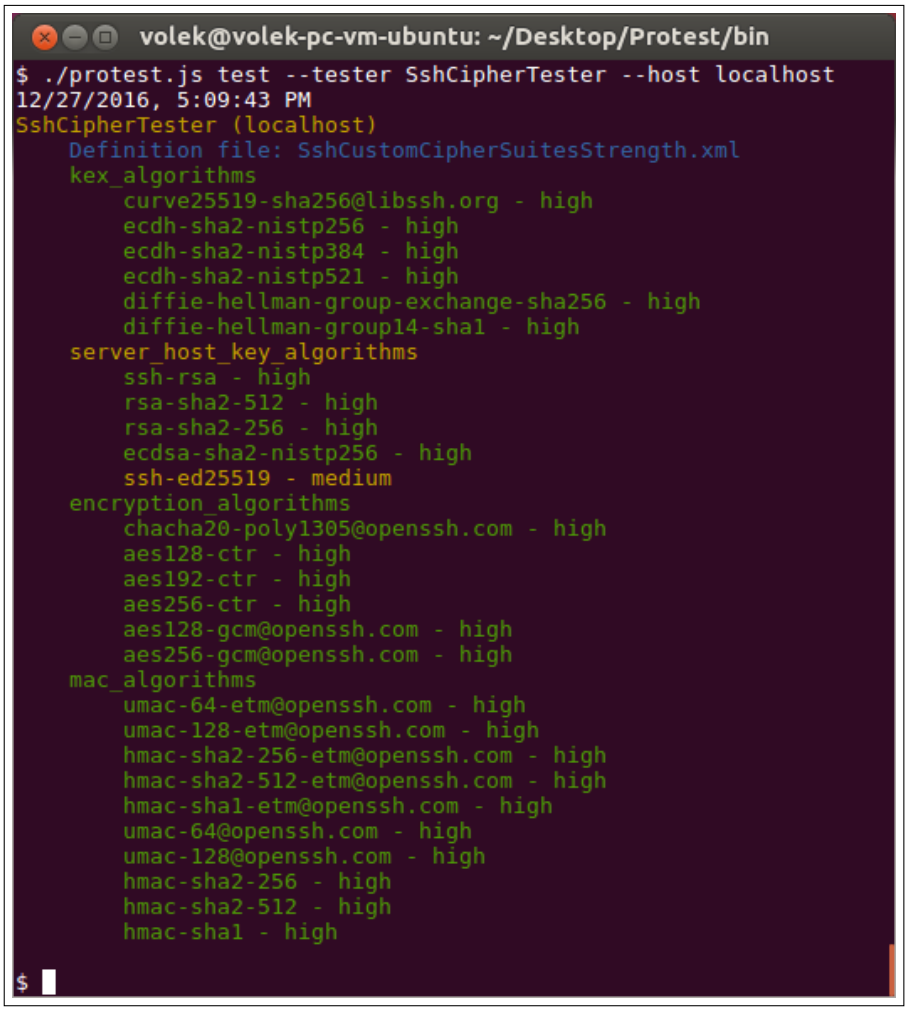
Tento testovací modul ověřuje certifikát SSL/TLS serveru. Ukázka běhu tohoto testu je na obrázku 3.2. Tento test probíhá v následujících krocích:

1. načtení informací o serverovém certifikátu, informace jsou získané pomocí příkazu `nmap -sV -sC`,
2. ověření časové platnosti certifikátu,
3. ověření shodnosti SHA1 hashe s očekávanou hashe z konfigurace testovacího modulu (parametrů).

3.3.3 `SshCipherTester`

Tento testovací modul testuje podporované algoritmy pro výměnu klíčů, autentizaci, šifrování a MAC. Ukázka běhu tohoto testu je na obrázku 3.3. Testování zde probíhá v následujících krocích:

1. zjištění podporovaných algoritmů jednotlivých částí šifrovací pomocí skriptu `ssh2-enum-algos` programu `nmap`,
2. načtení definičního souboru,
3. ohodnocení jednotlivých částí šifrovací sady dle definičního souboru.



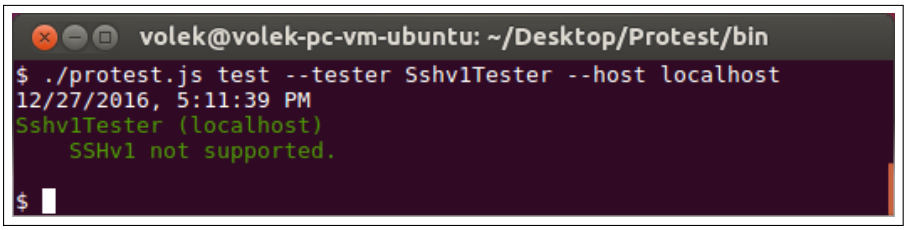
Obrázek 3.3: Běh testu `SshCipherTester` v příkazové řádce

3.3.4 `Sshv1Tester`

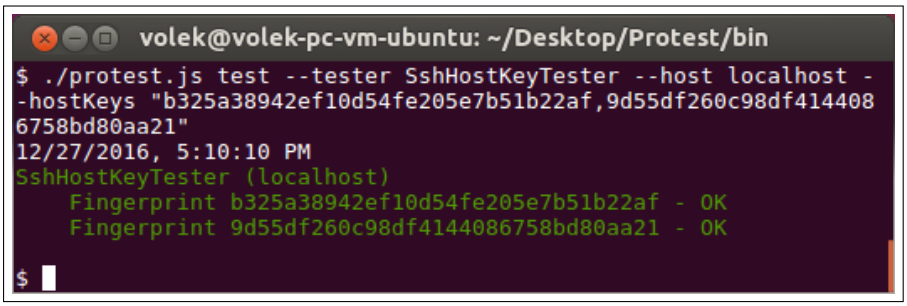
Tento testovací modul testuje, zda je podporována SSHv1. To je provedeno příkazem `nmap -sV -sC` a následným zpracováním výstupu. Ukázka běhu tohoto testu je na obrázku 3.4.

3.3.5 `SshHostKeyTester`

Tento testovací modul ověřuje shodnost otisků klíčů SSH serveru oproti konfiguraci testovacího modulu (parametrů). Otisky klíčů jsou získány pomocí skriptu `ssh-hostkey` programu `nmap` a následným zpracováním výstupu. Ukázka běhu tohoto testu je na obrázku 3.5.



Obrázek 3.4: Běh testu `Sshv1Tester` v příkazové řádce



Obrázek 3.5: Běh testu `SshHostKeyTester` v příkazové řádce

3.4 Spouštění testů

Jak je popsáno analýze v sekci 2.2.2, jsou implementovány tři možnosti spouštění testů:

- jednorázové spuštění konkrétního testu,
- jednorázové spuštění testů z konfigurace,
- opakované spuštění testů z konfigurace.

Každá z těchto možností je implementována pomocí tzv. runneru. Tedy hned po spuštění nástroje dojde k rozhodnutí, který runner se použije v závislosti na argumentech, se kterými byl program spuštěn. Existují tedy tyto tři runnery:

- **TestRunner** - jednorázové spuštění konkrétního testu,
- **OnceRunner** - jednorázové spuštění testů z konfigurace,
- **ScheduleRunner** - opakované spuštění testů z konfigurace.

3.4.1 Jednorázové spuštění konkrétního testu

Jednorázové spuštění konkrétního testu je možné z příkazové řádky. Je nutné specifikovat jméno konkrétního testeru, cílovou adresu a případně cílový port. Dále je nutné předat další parametry, pokud to konkrétní tester vyžaduje.

V tomto případě je konfigurační soubor ignorován. Tento způsob spouštění je určen spíše pro testovací a ladící účely. Příklady spuštění některých testů jsou uvedeny dále.

Příklad spuštění testu `SslTlsCipherTester`

```
$ prottest test --tester SslTlsCipherTester --host cvut.cz
        --evaluator OpenSSL
```

Příklad spuštění testu `SslTlsCertTester`

```
$ prottest test --tester SslTlsCertTester --host cvut.cz
        --checksum 9032b65d2eb903046cb5630392e281c3fbeccfde
```

Příklad spuštění testu `SshCipherTester`

```
$ prottest test --tester SshCipherTester --host localhost
```

3.4.2 Konfigurace pro automatizované spouštění testů

Nástroj Prottest je možné konfigurovat XML souborem. Příklad tohoto souboru je v ukázce souboru 4 na straně 47.

Sekce konfiguračního souboru

- `tests` - seznam testů,
- `httpServer` - nastavení HTTP serveru pro přístup k výstupům testů,
- `history` - jak dlouho se mají uchovávat výsledky,
- `smtpServer` - údaje o SMTP serveru určeného pro odesílání notifikací,
- `notification` - nastavení notifikací.

Konfigurační sekce `tests`

V konfiguračním souboru (viz ukázka 4) lze definovat sadu testů, která se bude spouštět najednou (viz sekce 3.4.3), případně opakovaně (viz sekce 3.4.4).

Do této sady testů bude administrátor doplňovat jednotlivé testy, které bude chtít opakovaně spouštět (například ve chvíli, kdy zprovozní nový emailový nebo SSH server).

Parametry testů

- `name` - jméno, určeno jen pro jednodušší identifikaci mezi výsledky,
- `tester` - název Testeru,
- `host` - cílová adresa,
- `port` - cílový port,
- další parametry - některé testy vyžadují či umožňují předání i dalších parametrů (například jméno konfiguračního souboru, který test využívá).

Konfigurační sekce `scheduler`

V sekci `scheduler` je atribut `cron`, který obsahuje cron výraz, podle kterého budou testy pravidelně spouštěny. Pokud tato sekce chybí, nebudou testy nijak automaticky spouštěny.

Konfigurační sekce `history`

V sekci `history` se určuje, kolik předchozích výsledků bude uchováno v úložišti výsledků a tedy kolik předchozích výsledků bude k dispozici přes webové rozhraní. Pokud tato sekce není uvedena, budou se uchovávat všechny výsledky.

Konfigurační sekce `httpServer`

V sekci `httpServer` je definován port, na kterém je spuštěn webový server, který zpřístupňuje výsledky dřívějších běhů testovacích modulů. Pokud tato sekce chybí, žádný server není spuštěn. Výsledky testů jsou pak přístupné jen v databázovém souboru.

Konfigurační sekce `smtpServer` a `notification`

V sekci `smtpServer` jsou přístupové údaje k SMTP serveru, který bude využit k odesílání emailů. V sekci `notification` jsou definovány adresy, ze které a na kterou mají být odeslány emaily s upozorněním na výsledek akce vyžadující pozornost správce. Pokud alespoň jedna sekce chybí, nebude docházet k žádnému odesílání emailů.

Důležitý je atribut `minimal` sekce `notification` - ten určuje, jaké výsledky testů vyžadují pozornost. Možnosti jsou stejné, jako možné výsledky testů, tedy: `ok`, `warning`, `critical`, `fail`.

3.4.3 Jednorázové spuštění testů z konfigurace

Pro otestování správné konfigurace lze spustit celou sadu testů definovanou v konfiguračním souboru. V tomto případě jsou výsledky testů vypsány na standardní výstup. Z konfigurace se nyní bere v potaz pouze sada testů (element `tests`), ostatní sekce jsou ignorovány.

Příklady jednorázového spuštění testu z konfiguračního souboru.

```
$ prottest once
```

```
$ prottest once -settings MySettings.xml
```

3.4.4 Opakované spuštění testů z konfigurace

Ve chvíli, když je vše připravené, lze spustit nástroj Prottest jako službu.

Při spuštění jako služby:

- spustí se plánovač, který pravidelně opakuje testy dle konfigurace,
- výsledky testů jsou ukládány do souboru,
- spustí se webový server, který umožňuje přístup k předchozím výsledkům přes webové rozhraní,
- v případě výsledků vyžadujících pozornost je odeslán upozorňující email dle konfigurace.

Příklady opakovaného spuštění testu z konfiguračního souboru

```
$ prottest run
```

```
$ prottest run --settings MySettings.xml
```

Ukázka souboru 4: Konfigurační soubor pro nástroj `Prottest`

```
<?xml version="1.0" encoding="UTF-8"?>
<protestSetttings>
    <scheduler cron="0 0 2 * * *" />

    <httpServer port="5000" />

    <history keep="20" />

    <smtpServer host="localhost" port="2525"
        user="" pass="" />

    <notification from="admin@protest.com"
        to="myemail@gmail.com" minimal="critical" />

    <tests>
        <test name="SSL OpenSSL cvut.cz"
            tester="SslTlsCipherTester"
            host="cvut.cz" port="443"
            evaluator="OpenSSL" />

        <test name="SSL Custom cvut.cz"
            tester="SslTlsCipherTester"
            host="cvut.cz" port="443"
            evaluator="Custom"
            evaluatorFile="SslCustomCipherSuitesStrengts.xml" />

        <test name="SSL Certificate google.com"
            tester="SslTlsCertTester" host="google.com"
            checksum="132c74e3cb4d308f4c18422edc8c9592e7e6b5a0" />

        <test name="SSH ciphers localhost"
            tester="SshCipherTester"
            host="localhost" port="22"
            evaluatorFile="SshCustomCipherSuitesStrength.xml" />
    </tests>
</protestSetttings>
```

Kapitola 4

Testování

4.1 Testován ve virtuální síti

Testování implementovaného nástroje jsem prováděl ve virtuální síti virtuálních strojů pomocí programu VirtualBox. Testování ve virtuální síti bylo vhodné, protože jsem mohl jednoduše instalovat různé služby a měnit jejich nastavení. Konkrétně se jednalo tři následující virtuální stroje:

- **Linux 1** - Ubuntu 16.04 - na tomto stroji běžel samotný nástroj Prottest,
- **Linux 2** - Ubuntu 16.04 - na tomto stroji běžely linuxové služby se šifrovanými protokoly, které byly testované nástrojem Prottest,
- **Windows** - Windows 7 x64 - na tomto stroji běžely Windows služby se šifrovanými protokoly, které byly testované nástrojem Prottest.

Všechny tři virtuální stroje měly dvě virtuální procesorová jádra a dedikované 4 GB RAM[30] paměti.

Stroj, na kterém virtualizované stroje běžely, měl konfiguraci: čtyřjádrový Intel Core i7-6700K (4 GHz), 32 GB RAM, Windows 10 x64.

V programu VirtualBox byla vytvořena virtuální síť, tzv. NAT Network. Do této sítě byly připojeny všechny tři virtuální stroje.

Na stroji Linux 1 musely být pro běh nástroje Prottest nainstalovány tyto programy: NodeJS, OpenSSL, nmap.

Implementované testovací moduly jsou navržené pro testování služeb postavených na SSL/TLS a SSH. Nainstaloval jsem nakonfiguroval jsem tyto služby v prostředí Linux a v prostředí Windows. Následně jsem nakonfiguroval nástroj Prottest tak, aby testoval správnou konfiguraci těchto služeb v těchto dvou prostředích.

[30] Random-access memory

Testování modulů pro SSL/TLS pro HTTPS

Na stanici Windows jsem nainstaloval a nakonfiguroval webový server IIS[31] a zprovoznil službu HTTPS se self-signed certifikátem. Poté jsem nakonfiguroval pomocí nástroje IISCrypto 2.0 [75] podporované šifrovací sady tak, aby byly podporovány šifrovací sady, které OpenSSL hodnotí stupněm high.

Poté jsem spustil nástroj Prottest jako službu. Výsledky testování podporovaných šifrovacích sad (testovací modul `SslTlsCipherTester`) byly v pořádku (všechno ve stavu ok).

Následně jsem změnil podporované šifrovací sady tak, aby byly povoleny i šifrovací sady, které jsou ohodnoceny jako low. Spuštěný nástroj Prottest na to ihned při dalším plánovaném běhu reagoval odesláním emailové notifikace, která obsahovala kompletní výsledky běhu testovacího modulu, při kterém byl nalezen bezpečností problém.

Dále jsem také testoval testovací modul `SslTlsCertTester`. Tento testovací modul byl nastaven, aby kontroloval SHA1 hash použitého certifikátu. Když SHA1 hash odpovídala, test proběhl úspěšně. Poté jsem zkusil ze stanice Linux 1 provést útok man in the middle pomocí ARP[32] poisoningu, při které byl podstrčen jiný certifikát.

Spuštěný nástroj Prottest na to opět při dalším plánovaném běhu reagoval odesláním emailové notifikace, která obsahovala kompletní výsledky běhu testovacího modulu, kde bylo patrné, že SHA1 hash neodpovídá očekávanému.

Testování modulů pro SSL/TLS pro emailové protokoly

Dále jsem na Windows nainstaloval emailový server hMailServer [76], který podporuje zabezpečené verze protokolů POP3, SMTP a IMAP. Tyto služby jsem nakonfiguroval se self-signed certifikátem.

Poté jsem nakonfiguroval Prottest na stanici Linux 1, aby pravidelně testoval konfiguraci POP3, SMTP a IMAP služeb na stanici s Windows (pomocí testovacích modulů `SslTlsCipherTester` a `SslTlsCertTester`). Následně jsem měnil nastavení emailových služeb pomocí administrátorské konzole k hMailServer. Prottest správně reagoval na změny nastavení služeb a v případě bezpečnostní hrozby odeslal email s upozorněním.

Testování modulů pro protokol SSH

Na stanici Linux 2 jsem nainstaloval SSH server pomocí balíčkovacího systému apt-get. Poté jsem spustil nástroj Prottest jako službu s nakonfigurovanými testovacími moduly `SshCipherTester`, `SshHostKeyTester` a `Sshv1Tester`.

[31] Internet Information Services
[32] Address Resolution Protocol

Následně jsem postupně měnil nastavení SSH serveru pomocí souboru `/etc/ssh/sshd_config`. Výsledky testů automaticky reagovaly na změnu konfigurace a v případě bezpečnostní hrozby byla odeslána emailová notifikace.

4.2 Testování v reálném prostředí

Dále jsem ještě testoval všechny testovací moduly oproti reálným službám provozovaných na internetu. U tohoto testování jsem neměl možnost měnit konfiguraci jednotlivých služeb, bylo ale ověřeno, že implementované testovací moduly fungují proti reálným službám.

Moduly pro SSL/TLS jsem testoval pro:

- HTTPS
  - `google.com:443`
  - `seznam.cz:443`
  - `cvut.cz:443`
- Emailové protokoly
  - `imap.gmail.com:993`
  - `smtp.gmail.com:465`
  - `smtp.gmail.com:587`
  - `imap.seznam.cz:993`
  - `pop3.seznam.cz:995`
  - `smtp.seznam.cz:465`

Moduly pro SSH jsem testoval pro:

- `fray1.fit.cvut.cz:22`
- `sshfreeshell.org:22`

Při tomto testování testovacího modulu `SslTlsCertTester` jsem narazil na problém, že někdy test doběhl a někdy skončil neočekávanou chybou při spouštění příkazu `nmap -sV -sC`. Po chvíli zkoumání jsem přišel na to, že doba běhu tohoto příkazu je poměrně dlouhá a někdy se příkaz předčasně ukončil z důvodu překročení časového limitu (timeoutu). Chybu jsem tedy opravil zvýšením časového limitu příkazu.

4.3 Závěry testování

Při testování jsem otestoval implementované testovací moduly. V průběhu tohoto finálního testování jsem našel jen jednu závažnou chybu (timeout nmap příkazu v testovacím modulu `SslTlsCertTester`). Ostatní chyby jsem odhalil už v době vývoje.

Důležitým závěrem testování je také to, že testy navržené pro rodinu protokolů SSL/TLS opravdu fungují i na protokoly jiné než HTTPS (konkrétně POP3, IMAP a SMTP). To se sice dalo očekávat, ale bylo vhodné to ověřit. V době vývoje jsem totiž testoval tyto moduly jen proti HTTPS serveru.

Závěr

Aktuální stav projektu

V této práci byly analyzovány a popsány jednotlivé verze častých síťových protokolů. Konkrétně byly popsány možnosti konfigurace a možné zranitelnosti protokolů SSL/TLS a SSH. Dále byly popsány některé VPN protokoly, konkrétně PPTP, L2TP/IPsec a OpenVPN.

Byl navrhnut a implementován nástroj pro automatizované testování správné bezpečné konfigurace služeb používající některé vybrané protokoly. Automatizované testování v tomto případě znamená pravidelně opakované testování bezpečného nastavení služeb. V případě nalezení potenciální bezpečnostního hrozby je správce upozorněn.

Součástí této práce jsou implementované testy protokolů SSL/TLS a SSH. Nástroj tedy lze použít na testování protokolů HTTPS, IMAP, SMTP, POP3, SSH a všech dalších protokolů postavených na SSL/TLS. Projekt je tedy v tuto chvíli použitelný pro testování výše uvedených protokolů.

Budoucnost projektu

Testovací nástroj je poměrně snadno rozšiřitelný. To znamená, že je možné přidávat jednotlivé testy pro různé protokoly a jejich zranitelnosti.

Do budoucna by tedy bylo vhodné implementovat například testy protokolu IPsec a dalších protokolů, které nejsou implementovány pomocí SSL/TLS nebo SSH. Případně je možné implementovat i testy pro nešifrované protokoly, které také mohou obsahovat zranitelnosti.

Osobní přínos

Přínos této práce pro mě byl především v jeho analytické části, kde jsem se detailně seznámil s jednotlivými verzemi protokolů a jejich bezpečnostními

zranitelnostmi a jinými problémy.

Dále jsem se také naučil pracovat s knihovnou `OpenSSL` a nástrojem `nmap`. Zároveň jsem se seznámil i s možností implementovat konzolové aplikace pomocí technologie NodeJS.

Zhodnocení dosažených výsledků

Osobně si myslím, že projekt byl pro mě přínosný a že byl úspěšný, protože byla provedena studie všech protokolů ze zadání a dále byl dle zadání navržen a implementován nástroj pro automatizované testování bezpečné konfigurace vybraných protokolů.

Za největší přínos považuji analytickou část, která může být užitečná jako zdroj informací při správném nastavování popisovaných služeb. Dále také může být využita jako zdroj informací při konfiguraci implementovaného nástroje pro automatizované testování.

Nástroj, který byl implementován, je robustní a je možné ho rozšiřovat. S aktuálně implementovanými testy lze automatizovaně testovat velkou část produkčních či interních služeb či aplikací na různých serverech a lze ho tak využít v reálném prostředí.

Literatura

[1] Hack Like a Pro Web Site. [cit. 2016-08-01]. Dostupné z: http://null-byte.wonderhowto.com/how-to/hack-like-a-pro/

[2] Emre Durdağı, A. B.: IPV4/IPV6 security and threat comparisons. 2010, [cit. 2016-08-01]. Dostupné z: http://www.sciencedirect.com/science/article/pii/S187704281000902X

[3] Shuler, R.: How Does the Internet Work? 2002, [cit. 2016-04-01]. Dostupné z: https://web.stanford.edu/class/msande91si/www-spr04/readings/week1/InternetWhitepaper.htm

[4] Rouse, M.: Man-in-the-middle attack (MitM). [cit. 2016-08-01]. Dostupné z: http://internetofthingsagenda.techtarget.com/definition/man-in-the-middle-attack-MitM

[5] Conti, M.; Dragoni, N.; Lesyk, V.: A Survey of Man In The Middle Attacks. *IEEE Communications Surveys Tutorials*, ročník 18, č. 3, thirdquarter 2016: s. 2027–2051, ISSN 1553-877X, doi:10.1109/COMST.2016.2548426.

[6] Callegati, F.; Cerroni, W.; Ramilli, M.: Man-in-the-Middle Attack to the HTTPS Protocol. *IEEE Security Privacy*, ročník 7, č. 1, Jan 2009: s. 78–81, ISSN 1540-7993, doi:10.1109/MSP.2009.12.

[7] Delfs, H.; Knebl, H.: *Introduction to Cryptography: Principles and Applications*. Information Security and Cryptography, Springer Berlin Heidelberg, 2012, ISBN 9783642871269. Dostupné z: https://books.google.cz/books?id=DOWpCAAAQBAJ

[8] Wikipedia: Public-key cryptography. 2016, [cit. 2016-08-01]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Public-key_cryptography&oldid=753129410

[9] García, D. F.: Performance Evaluation of Advanced Encryption Standard Algorithm. In *2015 Second International Conference on Mathematics and Computers in Sciences and in Industry (MCSI)*, Aug 2015, s. 247–252, doi:10.1109/MCSI.2015.61.

[10] Clercq, J. D.: Symmetric vs. Asymmetric Ciphers. 2006, [cit. 2016-08-01]. Dostupné z: http://windowsitpro.com/security/symmetric-vs-asymmetric-ciphers

[11] Moriarty, K. M.; aj.: PKCS #1: RSA Cryptography Specifications Version 2.2. RFC 8017, IETF, November 2016, [cit. 2016-08-01]. Dostupné z: https://tools.ietf.org/html/rfc8017

[12] Laboratories, R.: RSA CRYPTOGRAPHY STANDARD. 2002, [cit. 2016-08-01]. Dostupné z: https://www.emc.com/emc-plus/rsa-labs/standards-initiatives/pkcs-rsa-cryptography-standard.htm

[13] Rescorla, E.: Diffie-Hellman Key Agreement Method. RFC 2631, IETF, June 1999, [cit. 2016-08-01]. Dostupné z: https://tools.ietf.org/html/rfc2631

[14] DNSsimple: What is the SSL Certificate Chain? [cit. 2016-08-01]. Dostupné z: https://support.dnsimple.com/articles/what-is-ssl-certificate-chain/

[15] Wikipedia: Chain of trust. 2016, [cit. 2016-08-01]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Chain_of_trust&oldid=752275642

[16] Arora, M.: How secure is AES against brute force attacks? 2012, [cit. 2016-08-01]. Dostupné z: http://www.eetimes.com/document.asp?doc_id=1279619

[17] Khan, S.; Khan, F.: Attempt based password. In *2016 13th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, Jan 2016, s. 300–304, doi:10.1109/IBCAST.2016.7429894.

[18] Dworkin, M.: Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms - NIST 800-38B. 2005, [cit. 2016-08-01]. Dostupné z: https://doi.org/10.6028/NIST.SP.800-38B

[19] ISO: ISO/IEC 7498-1:1994. 1994, [cit. 2016-08-01]. Dostupné z: http://www.iso.org/iso/catalogue_detail.htm?csnumber=20269

[20] Braden, R.: Requirements for Internet Hosts – Communication Layers. RFC 1122, IETF, October 1989, [cit. 2016-08-01]. Dostupné z: https://tools.ietf.org/html/rfc1122

[21] Braden, R.: Requirements for Internet Hosts – Application and Support. RFC 1123, IETF, October 1989, [cit. 2016-08-01]. Dostupné z: https://tools.ietf.org/html/rfc1123

[22] Stephen Kent, K. S.: Security Architecture for the Internet Protocol. RFC 4301, IETF, December 2005, [cit. 2016-08-01]. Dostupné z: https://tools.ietf.org/html/rfc4301

[23] Barbulescu, R.; Gaudry, P.; Joux, A.; aj.: A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. Cryptology ePrint Archive, Report 2013/400, 2013, [cit. 2016-08-01]. Dostupné z: http://eprint.iacr.org/2013/400

[24] Wikipedia: Diffie–Hellman key exchange. 2016, [cit. 2016-08-01]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Diffie%E2%80%93Hellman_key_exchange&oldid=756216049

[25] Elaine Barker, A. R. M. S., Lily Chen: Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography - NIST 800-56A Rev. 2. 2013, [cit. 2016-08-01]. Dostupné z: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar2.pdf

[26] Brown, D. R. L.: Elliptic Curve Cryptography. 2009, [cit. 2016-08-01]. Dostupné z: http://www.secg.org/sec1-v2.pdf

[27] Zákon č. 181/2014 Sb. o kybernetické bezpečnosti a o změně souvisejících zákonů (zákon o kybernetické bezpečnosti). 2014, [cit. 2016-08-01]. Dostupné z: https://www.nbu.cz/cs/pravni-predpisy/zakon-o-kyberneticke-bezpecnosti-a-o-zmene-souvisejicich-zakonu-zakon-o-kyberneticke-bezpecnosti/

[28] Prováděcí právní předpisy k zákonu č. 181/2014 Sb., o kybernetické bezpečnosti a o změně souvisejících zákonů (zákon o kybernetické bezpečnosti). 2014, [cit. 2016-08-01]. Dostupné z: https://www.nbu.cz/cs/pravni-predpisy/provadeci-pravni-predpisy/provadeci-pravni-predpisy-k-zakonu-c-1812014-sb-o-kyberneticke-bezpecnosti-a-o-zmene-souvisejicich-zakonu/

[29] Sean Turner, T. P.: Prohibiting Secure Sockets Layer (SSL) Version 2.0. RFC 6176, IETF, 2011, [cit. 2016-08-01]. Dostupné z: https://tools.ietf.org/html/rfc6176

[30] Alan O. Freier, P. C. K.: Deprecating Secure Sockets Layer Version 3.0. RFC 6101, IETF, 2011, [cit. 2016-08-01]. Dostupné z: https://tools.ietf.org/html/rfc6101

[31] Sean Turner, T. P.: Deprecating Secure Sockets Layer Version 3.0. RFC 7568, IETF, 2015, [cit. 2016-08-01]. Dostupné z: https://tools.ietf.org/html/rfc7568

[32] Tim Dierks, A. O. F.: The TLS Protocol Version 1.0. RFC 2246, IETF, 1999, [cit. 2016-08-01]. Dostupné z: https://tools.ietf.org/html/rfc2246

[33] Win Treese, E. R.: The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346, IETF, 2006, [cit. 2016-08-01]. Dostupné z: https://tools.ietf.org/html/rfc4346

[34] Win Treese, E. R.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, IETF, 2008, [cit. 2016-08-01]. Dostupné z: https://tools.ietf.org/html/rfc5246

[35] Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3 (draft). RFC, IETF, 2016, [cit. 2016-08-01]. Dostupné z: https://tools.ietf.org/html/draft-ietf-tls-tls13-18

[36] wolfSSL: Differences between SSL and TLS Protocol Versions. 2010, [cit. 2016-08-25]. Dostupné z: https://www.wolfssl.com/wolfSSL/Blog/Entries/2010/10/7_Differences_between_SSL_and_TLS_Protocol_Versions.html

[37] Appel, S.; aj.: What is the difference between SSL 2.0 and 3.0? [cit. 2016-08-25]. Dostupné z: http://stason.org/TULARC/security/ssl-talk/4-11-What-is-the-difference-between-SSL-2-0-and-3-0.html

[38] lei Zhang, H.: Three attacks in SSL protocol and their solutions. [cit. 2016-09-07]. Dostupné z: https://www.cs.auckland.ac.nz/courses/compsci725s2c/archive/termpapers/725zhang.pdf

[39] US-CERT: Alert (TA14-290A), SSL 3.0 Protocol Vulnerability and POODLE Attack. 2014, [cit. 2016-09-07]. Dostupné z: https://www.us-cert.gov/ncas/alerts/TA14-290A

[40] Beattie, D.: POODLE Vulnerability Expands Beyond SSLv3 to TLS 1.0 and 1.1. 2014, [cit. 2016-09-07]. Dostupné z: https://www.globalsign.com/en/blog/poodle-vulnerability-expands-beyond-sslv3-to-tls/

[41] Wikipedia: Transport Layer Security. 2016, [cit. 2016-09-07]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Transport_Layer_Security&oldid=737077715

[42] OpenSSL Web Site. [cit. 2016-08-01]. Dostupné z: https://www.openssl.org/

[43] OpenSSL: OpenSSL ciphers command documentation. [cit. 2016-08-01]. Dostupné z: https://www.openssl.org/docs/man1.0.2/apps/ciphers.html

[44] Ylonen, T.; Lonvick, C.: The Secure Shell (SSH) Authentication Protocol. RFC 4252, IETF, January 2006, [cit. 2016-04-01]. Dostupné z: https://tools.ietf.org/html/rfc4252

[45] US-CERT: Vulnerability Note VU#684820, SSH-1 allows client authentication to be forwarded by a malicious server to another server. 2001, [cit. 2016-04-01]. Dostupné z: https://www.kb.cert.org/vuls/id/684820

[46] NIST: Vulnerability Summary for CVE-2001-1473. 2001, [cit. 2016-04-01]. Dostupné z: https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2001-1473

[47] Tatu Ylonen, C. L.: The Secure Shell (SSH) Transport Layer Protocol. RFC 4253, IETF, 2006, [cit. 2016-08-01]. Dostupné z: https://tools.ietf.org/html/rfc4253

[48] Tatu Ylonen, C. L.: The Secure Shell (SSH) Connection Protocol. RFC 4254, IETF, 2006, [cit. 2016-08-01]. Dostupné z: https://tools.ietf.org/html/rfc4254

[49] Jakob Schlyter, W. G.: Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints. RFC 4255, IETF, 2006, [cit. 2016-08-01]. Dostupné z: https://tools.ietf.org/html/rfc4255

[50] NIST: Guide to SSL VPNs, NIST 800-113. 2008, [cit. 2016-08-01]. Dostupné z: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-113.pdf

[51] NIST: Guide to IPsec VPNs, NIST 800-77. 2005, [cit. 2016-08-01]. Dostupné z: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-77.pdf

[52] Kory Hamzeh, G. S. P.: Point-to-Point Tunneling Protocol (PPTP). RFC 2637, IETF, 1999, [cit. 2016-08-01]. Dostupné z: https://tools.ietf.org/html/rfc2637

[53] Chirgwin, R.: Marlinspike demos MS-CHAPv2 crack. 2012, [cit. 2016-04-01]. Dostupné z: http://www.theregister.co.uk/2012/07/31/ms_chapv2_crack/

[54] Pick, V.: VPN Newbie Guide: Picking between OpenVPN, PPTP and L2TP. 2014, [cit. 2016-04-01]. Dostupné z: http://vpnpick.com/vpn-newbie-guide-picking-openvpn-pptp-l2tp/

[55] openvpn: Security Overview - OpenVPN cryptographic layer. [cit. 2016-04-01]. Dostupné z: https://openvpn.net/index.php/open-source/documentation/security-overview.html

[56] Pall, G. S.; aj.: Layer Two Tunneling Protocol L2TP. RFC 2661, IETF, August 1999, [cit. 2016-04-01]. Dostupné z: https://tools.ietf.org/html/rfc2661

[57] Jankiewicz, E.; aj.: IPv6 Node Requirements. RFC 6434, IETF, December 2011, [cit. 2016-04-01]. Dostupné z: https://tools.ietf.org/html/rfc6434

[58] Stephen Kent, S. K.: Security Architecture for the Internet Protocol. RFC 2401, IETF, 1998, [cit. 2016-08-01]. Dostupné z: https://tools.ietf.org/html/rfc2401

[59] Rodney Thayer, N. D.: IP Security Document Roadmap. RFC 2411, IETF, 1998, [cit. 2016-08-01]. Dostupné z: https://tools.ietf.org/html/rfc2411

[60] David McGrew, P. H.: Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH). RFC 7321, IETF, 2014, [cit. 2016-08-01]. Dostupné z: https://tools.ietf.org/html/rfc7321

[61] Bouška, P.: VPN 1 - IPsec VPN a Cisco. 2011, [cit. 2016-04-01]. Dostupné z: http://www.samuraj-cz.com/clanek/vpn-1-ipsec-vpn-a-cisco/

[62] Charlie Kaufman, P. H.: Internet Key Exchange Protocol Version 2 (IKEv2). RFC 7296, IETF, 2014, [cit. 2016-08-01]. Dostupné z: https://tools.ietf.org/html/rfc7296

[63] nmap Scripts. [cit. 2016-08-01]. Dostupné z: https://nmap.org/nsedoc/

[64] npm Web Site. [cit. 2016-08-01]. Dostupné z: https://www.npmjs.com/

[65] GNU Bash Web Site. [cit. 2016-08-01]. Dostupné z: https://www.gnu.org/software/bash/

[66] Python Web Site. [cit. 2016-08-01]. Dostupné z: https://www.python.org/

[67] .NET Core Web Site. [cit. 2016-08-01]. Dostupné z: https://www.microsoft.com/net/core

[68] NodeJS Web Site. [cit. 2016-08-01]. Dostupné z: https://nodejs.org/

[69] Wikipedia: Node.js. 2017, [cit. 2016-08-01]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Node.js&oldid=758795594

[70] TypeScript Web Site. [cit. 2016-08-01]. Dostupné z: https://www.typescriptlang.org/

[71] Visual Studio Code Web Site. [cit. 2016-08-01]. Dostupné z: https://code.visualstudio.com/

[72] gulp.js Web Site. [cit. 2016-08-01]. Dostupné z: http://gulpjs.com/

[73] Mak Kolybabi, G. L.: Nmap script - ssl-enum-ciphers. [cit. 2016-08-01]. Dostupné z: https://nmap.org/nsedoc/scripts/ssl-enum-ciphers.html

[74] nmap Web Site. [cit. 2016-08-01]. Dostupné z: https://nmap.org/

[75] IIS Crypto Web Site. [cit. 2016-08-01]. Dostupné z: https://www.nartac.com/Products/IISCrypto

[76] hMailServer Web Site. [cit. 2016-08-01]. Dostupné z: https://www.hmailserver.com/

Příloha **A**

Seznam použitých zkratek

3DES Triple Data Encryption Standard

ACE-KEM Asymetric Ciphers and Key Encapculation Mechanism

AES Advanced Encryption Standard

AH Authentication Header

API Application programming interface

ARP Address Resolution Protocol

Bash Bourne Again Shell

CBC Cipher Block Chaining

CBC Cipher Block Chaining

CCM Counter with CBC-MAC

CFB Cipher Feedback

CTR Counter Mode

DES Data Encryption Standard

DH Diffie-Hellman

DNS Domain Name System

DNSSEC Domain Name System Security Extensions

DSA Digital Signature Algorithm

DSS Digital Signature Standard

ECDH Elliptic Curve Diffie-Hellman

ECDSA Elliptic Curve Digital Signature Algorithm

ECIES-KEM Elliptic Curve Integrated Encryption System - Key Encapculation Mechanism

ESP Encapsulating Security Payload

FIPS Federal Information Processing Standards

FTP File Transfer Protocol

HMAC Keyed-Hash Message Authentication Code

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

IEC International Electrotechnical Commission

IKE Internet Key Exchange

IMAP Internet Message Access Protocol

IP Internet Protocol

IPsec Internet Protocol Security

ISAKMP Internet Security Association and Key Management Protocol

ISO International Organization for Standardization

L2TP Layer 2 Tunneling Protocol

MAC Message Authentication Code

MitM Man in the Middle

MS-CHAP Microsoft Challenge-Handshake Authentication Protocol

NAT Network address translation

NIST National Institute of Standards and Technology

OCB Offset Codebook Mode

OFB Output Feedback

OSI Open Systems Interconnection model

POP3 Post Office Protocol (version 3)

PPTP Point-to-Point Tunneling Protocol

PSEC-KEM Provably Secure Elliptic Curve - Key Encapculation Mechanism

RAM Random-access memory

RFC Request for Comments

RIPEMD RACE Integrity Primitives Evaluation Message Digest

RSA Rivest-Shamir-Adleman cryptosystem

RSA-KEM Rivest Shamir Adleman - Key Encapculation Mechanism

RSA-OAEP Rivest Shamir Adleman - Optimal Asymetric Encryption Pedding

RSA-PSS Rivest-Shamir-Adleman Probablistic Signature Scheme

SA Security Association

SCP Secure copy

SFTP Secure File Transfer Protocol

SHA Secure Hash Algorithm

SMTP Simple Mail Transfer Protocol

SSH Secure Shell

SSHFP SSH Fingerprint

SSL Secure Sockets Layer

SSTP Secure Socket Tunneling Protocol

TCP Transmission Control Protocol

TSL Transport Layer Security

UDP User Datagram Protocol

VPN Virtual Private Network

XML Extensible Markup Language

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `bin` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `zadani.pdf` ..... zadání práce ve formátu PDF