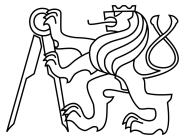
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Nástroj na testování zranitelností na běžné síťové útoky
Student:	Bc. Pavel Soukup
Vedoucí:	Ing. Tomáš Zahradnický, Ph.D.
Studijní program:	Informatika
Studijní obor:	Počítačová bezpečnost
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Nastudujte a popište útoky Shellshock, POODLE, CRIME a BEAST, případně další, po dohodě s vedoucím práce. Navrhněte a vytvořte vlastní nástroj pro příkazovou řádku platformy GNU/Linux, který bude testovat zranitelnost uživatelem zadaného serveru vůči těmto útokům. Nástroj koncipujte tak, aby z výpisů, které bude generovat, bylo možné zrekonstruovat jeho komunikaci s testovaným serverem v okamžiku testování.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

prof. Ing. Róbert Lórencz, CSc.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 2. února 2016

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

Diplomová práce

Nástroj na testování zranitelností na běžné síťové útoky

Bc. Pavel Soukup

Vedoucí práce: Ing. Tomáš Zahradnický, Ph.D.

8. května 2016

Poděkování

Tímto bych chtěl velice poděkovat panu Ing. Tomáši Zahradnickému Ph.D. za ochotu být vedoucím mé diplomové práce a za jeho pomoc při řešení problémů, které se vyskytly při vypracovávání této práce. Nemenší díky patří mým rodičům, kteří mi umožnili studovat, a snad i vystudovat ČVUT v Praze. V neposlední řadě děkuji své přítelkyni, její rodině a svým nejbližším příbuzným za dodání energie, elánu a podporu po celou dobu studia.

Prohlášení

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

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

V Praze dne 8. května 2016 .....

Odkaz na tuto práci

Soukup, Pavel. *Nástroj na testování zranitelností na běžné síťové útoky.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Tato diplomová práce se věnuje návrhu a implementaci nástroje na testování zranitelnosti uživatelem zadaného serveru vůči síťovým útokům BEAST, CRIME, POODLE, Lucky13 a Shellshock. V první části je popsána rodina protokolů SSL/TLS. Popis je zaměřen především na protokol SSL 3.0. Následně jsou popsány útoky na protokoly SSL/TLS BEAST, CRIME, POODLE a Lucky13. Spolu s nimi je zde popsán také útok Shellshock. Druhá část této práce se věnuje návrhu a implementaci rozšiřujících modulů pro program SSLyze a aplikaci na testování zranitelnosti serveru vůči útoku Shellshock.

Klíčová slova protokoly SSL/TLS, útok BEAST, útok CRIME, útok POODLE, útok Lucky13, útok Shellshock, rozšiřující moduly pro SSLyze, aplikace na testování zranitelnosti serveru

Abstract

This thesis deals with design and realization of tool for testing user specified server to vulnerability on network attacks BEAST, CRIME, POODLE, Lucky13 and Shellshock. There are description of SSL/TLS protocols, especially protocol SSL 3.0. Further there are descriptions of attack against SSL/TLS protocols, which are BEAST, CRIME, POODLE and Lucky13. Also Shellshock attack is decribed with them here. The second part of my thesis contains design and realization of plugins for SSLyze and application for testing vulnerability to Shellshock attack.

Keywords SSL/TLS protocols, BEAST attack, CRIME attack, POODLE attack, Lucky13 attack, Shellshock attack, plugins for SSLyze, application to tests server vulnerability

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Tato diplomová práce se bude zabývat bezpečností serverů vůči běžným síťovým útokům. Budou popsány některé ze současných běžných síťových útoků na server. Po popsání těchto útoků budou navrženy nástroje v jazyce pro příkazovou řádku platformy UNIX/Linux, které budou testovat zranitelnost uživatelem zadaného serveru vůči těmto útokům. Nástroje budou koncipovány tak, aby bylo z výpisu možné zrekonstruovat jejich komunikaci se serverem v okamžiku testování.

Nejprve bude popsána rodina protokolů SSL/TLS. Bude zmíněna její historie a její pozice v síťové komunikaci. Popis bude zaměřen na protokol SSL 3.0, který sice není v současné době považován za bezpečný, ale vychází z něho novější protokoly TLS. Dalším důvodem pro to je, že většina síťových útoků je vedena právě proti SSL 3.0. V popisu protokolu SSL 3.0 bude uvedena jeho základní struktura a budou popsány jeho součásti.

Dále bude analytická část této diplové práce obsahovat popis běžných síťových útoků. Budou popsány útoky BEAST, CRIME, POODLE, Lucky13 a Shellshock. V rámci popisu jednotlivých útoků budou zmíněny základní informace jako jsou kdo útok prezentoval a kdy byl útok prezentován. Také bude popsán princip útoku a zranitelnost, které daný útok využívá.

Nakonec analytické části budou uvedeny některé programy, které jsou v současné době dostupné pro otestování serveru na známé zranitelnosti. Podrobněji bude popsán program SSLyze, který bude používán v dalších částí této diplomové práce. Popsána bude především struktura a fungování programu s ohledem na jeho možné rozšíření o zásuvné moduly.

Na základě analýzi jednotlivých útoků budou navrženy testovací nástroje. Z hlediska lepší využitelnosti v praxi bylo po dohodě s vedoucím domluveno, že testovací nástroje budou mít formu rozšiřujících modulů do již existujícího a používaného programu pro testování. Tyto zásuvné moduly budou testovat zranitelnosti uživatelem zadaného serveru na útoky BEAST, CRIME, POODLE a Lucky13. Zásuvné moduly budou rozšiřovat funkčnost programu SSLyze. Jako samostatná aplikace bude navržen nástroj pro testování serveru

na zranitelnost vůči útoku Shellshock.

Navržené zásuvné moduly a aplikace budou implementovány. Implementace bude provedena, tak aby zásuvné moduly byly součástí programu SSLyze, pro který budou navrhovány. Budou zachovány všechny současné funkcionality programu a z výpisu implementovaných zásuvných modulů bude možné zrekonstruovat komunikaci se serverem v okamžiku testování. Aplikace bude implementována tak, aby z výpisu bylo jasné zda je zadaný server zranitelný, či nikoliv.

Kapitola 2

Analýza

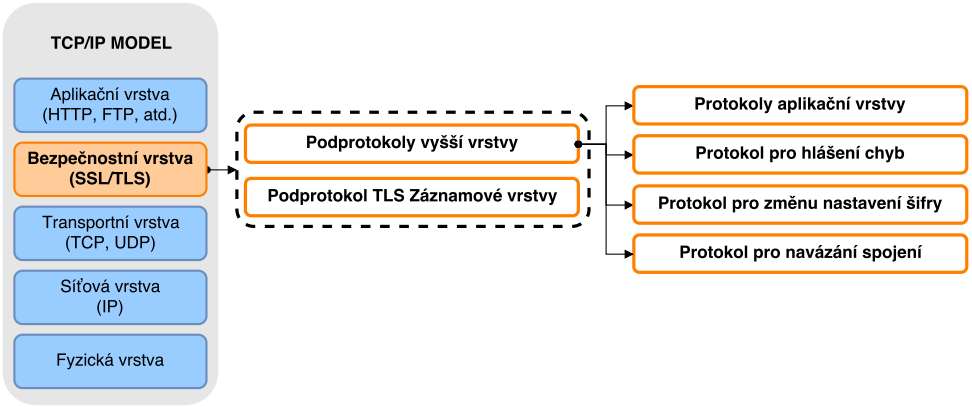
2.1 Protokol SSL a TLS

V této sekci jsou popsány protokoly SSL verze 3.0 [1, 2, 3] a TLS [4, 5, 6]. Protokol TLS vychází z protokolu SSL verze 3.0. Ačkoliv jsou si protokol SSL verze 3.0 a TLS verze 1.0 velice blízké, tak nemůže komunikovat TLS klient se SSL serverem a naopak. Pro základní pochopení fungování bezpečnostních protokolů je v této sekci vycházeno především z popisu protokolu SSL verze 3.0.

Oba protokoly slouží pro zajištění bezpečné komunikace mezi dvěma aplikacemi. Rodina protokolů SSL byla vytvořena společností Netscape a první verzí používanou na internetu byla verze 2.0. SSL protokol verze 3.0 byl představen v roce 1996. Nástupcem a aktuálním bezpečnostním standardem na internetu se stal v roce 1999 protokol TLS verze 1.0. V současnosti je doporučováno používání verze 1.2 tohoto protokolu.

Protokoly TLS a SSL vytváří další vrstvu v síťových modelech, vizte obr. 2.1. Jedná se o bezpečnostní vrstvu, která je v upraveném modelu TCP/IP umístěna mezi aplikační vrstvu a vrstvu transportní. Její úlohou je zabezpečit data aplikační vrstvy a předat je transportnímu protokolu TCP.

Samotný protokol SSL/TLS můžeme rozdělit do dvou vrstev, jak je to vyobrazeno na obr. 2.1. Tou nejnižší vrstvou je vrstva nazvaná jako Podprotokol TLS Záznamové vrstvy. Tato vrstva zajišťuje bezpečnou komunikaci mezi klientem a serverem pomocí Protokolu pro záznamovou vrstvu (RLP). Úkolem RLP protokolu je zapouzdření různých zpráv od vyšší vrstvy protokolu SSL/TLS. Tato vrsta se nazývá Podprotokoly vyšší vrstvy, používá podprotokoly SSL/TLS a přebírá data od aplikační vrstvy. Vrstva Podprotokolů vyšší vrstvy využívá Protokol pro navázání spojení (HP), Protokol pro změnu nastavení šifry (CCSP), který slouží pro potvrzení nastavení bezpečného spojení mezi klientem a serverem, a Protokol pro hlášení chyb (AP), pomocí kterého zasílá hlášení o chybách, které se vyskytly během navazování spojení. Hlavním úkolem této vrstvy je navázání bezpečného spojení mezi klientem a serverem.



Obrázek 2.1: Zobrazení umístění SSL/TLS vrstvy v modelu TCP/IP a struktura SSL/TLS vrstvy.

Klient i server si udržují některé parametry komunikace ve svých datových strukturách dvakrát. Jednou si udržují aktuální data používaná protokolem RLP a podruhé připravovaná data protokolem HP.

Klient se serverem si pro komunikaci zřizují tzv. relace. Tato relace zahrnuje jedno nebo více spojení, a mezi klientem a serverem může být současně zřízeno i více relací. Pro oboje si obě strany uchovávají následující informace ve svých strukturách:

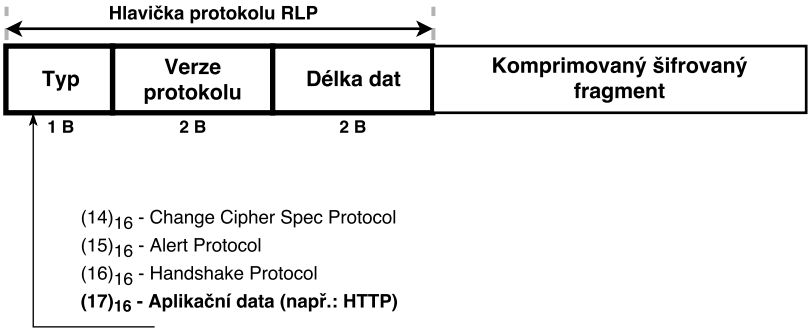
- pro relaci
  - Identifikační číslo relace
  - Certifikát druhé strany, je-li k dispozici
  - Identifikátor komprimačního algoritmu
  - Šifrovací sadu. Ta určuje algoritmus symetrického šifrování a algoritmus pro výpočet kontrolního součtu
  - Sdílené tajemství, 48 B známých pouze oběma stranám komunikace
  - Příznak obnovitelnosti relace
- pro spojení
  - Náhodné číslo generované klientem (nonce klienta)
  - Náhodné číslo generované serverem (nonce serveru)
  - Tajemství pro výpočet kontrolního součtu používané serverem
  - Tajemství pro výpočet kontrolního součtu používané klientem
  - Symetrický šifrovací klíč serveru
  - Symetrický šifrovací klíč klienta

– Inicializační vektor
– Číslo přijaté a odeslané zprávy

2.1.1 Protokol záznamové vrstvy

Protokol záznamové vrstvy, anglicky označován jako Record Layer Protocol, přebírá data od aplikačních protokolů, AP protokolu, CCSP protokolu a HP protokolu. Data od těchto vyšších protokolů jsou před předáním protokolu TCP upravena následujícím postupem:

1. Data jsou rozdělena na fragmenty o délce maximálně 2^{14} B.
2. Je provedena komprimace dat, pokud se na ní klient se serverem dohodli.
3. Vypočte se kontrolní součet dohodnutým algoritmem.
4. Fragment je doplněn na délku, která je násobkem délky šifrovacího bloku.
5. Zašifruje se fragment.
6. Je doplněna hlavička RLP protokolu před datový fragment.

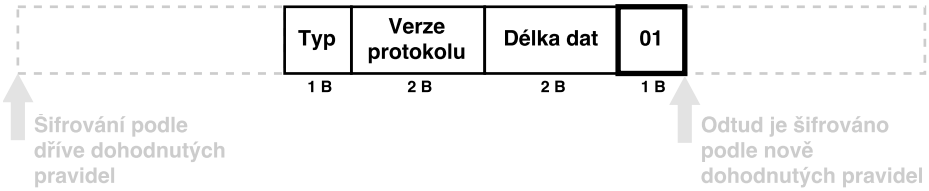


Obrázek 2.2: Formát paketu protokolu RLP [4].

Hlavička protokolu RLP je dlouhá 5 bajtů a rozdělena do třech polí, jak je vidět na obrázku 2.2. Hodnota v typu specifikuje, zda jsou přenášena data nějakého aplikačního protokolu (např.: HTTP), nebo se jedná o data některého z podprotokolů protokolu SSL/TLS, tedy protokol HP, CCSP, nebo AP. Následují 2 bajty, z nichž první bajt určuje verzi a druhý její detail. V praxi se setkáváme s těmito hodnotami: $(20)_{16}$ pro SSL verze 2, $(30)_{16}$ pro SSL verze 3, $(31)_{16}$ pro TLS verze 1, $(32)_{16}$ pro TLS verze 1.1 a $(33)_{16}$ pro TLS verze 1.2. Poslední dva bajty hlavičky udávají délku fragmentu po případné komprimaci a zašifrování.

2.1.2 Protokol pro změnu nastavení šifry

Protokol pro změnu nastavení šifry, který je v angličtině označován jako Change Cipher Spec Protocol (CCSP), slouží k informování o změně šifrovací strategie. Protokol se skládá z jedné zprávy, která je šifrována a komprimována aktuální šifrovací sadou. Zpráva obsahuje jeden bajt s hodnotou `1`. Tato zpráva je poslední, která je šifrována aktuální šifrovací sadou. Všechny následné zprávy jsou již šifrovány novou šifrovací sadou.

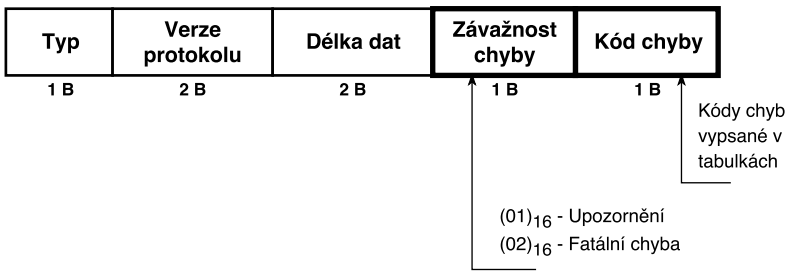


Obrázek 2.3: Formát paketu protokolu CCSP [4].

Zpráva o změně šifrové sady je posílána oběma stranami, klientem i serverem, nezávisle na sobě, aby oznámili přijetí změn, že následující zprávy budou chráněny dohodnutou šifrovací sadou a klíči. Klient posílá zprávu o změně specifikace šifrování následovanou zprávami o výměně klíčů a ověření certifikátu. Server posílá jednu zprávu po úspěšném zpracování zprávy o výměně klíčů, kterou obdržel od klienta. Neočekávaná zpráva o změně šifrovací sady může vyvolat varovnou zprávu `unexpected_message`, která je zaslána protokolem AP. Pokud se obnovuje předchozí relace, je zpráva o změně šifrovací sady odeslána po úvodní zprávě.

2.1.3 Protokol pro hlášení chyb

Tento protokol, anglicky označován jako Alert Protocol, zasílá chyby, problémy nebo varování ohledně spojení mezi oběma stranami. Zpráva je tvořena dvěma poli. První udává úroveň závažnosti vzniklé chyby a druhá pole specifikuje typ vzniklé chyby.



Obrázek 2.4: Formát paketu protokolu AP [4].

2.1.3.1 Úroveň závažnosti chyby

Toto pole obsahuje hodnotu `1` nebo `2` v závislosti na úrovni závažnosti vzniklé chyby. Zpráva s hodnotou `1` je výstražná nebo upozorňující zpráva a spojení v tomto případě může pokračovat. Zpráva s hodnotou `2` je zpráva označující fatální chybu a spojení je okamžitě ukončeno. V tomto případě však ostatní spojení, patřící k dané relaci, mohou pokračovat, ale identifikátor relace musí být nepřípustný pro vytvoření nového spojení.

Tabulka 2.1: Hodnoty v poli specifikace typu chyby u AP protokolu SSL 3.0 [1].

Název chyby	**Kód chyby**
CloseNotify	`0x00`
BadCertificate	`0x2A`
UnexpectedMessage	`0x0A`
UnsupportedCertificate	`0x2B`
BadRecordMAC	`0x14`
CertificateRevoked	`0x2C`
DecompressionFailure	`0x1E`
CertificateExpired	`0x2D`
HandshakeFailure	`0x28`
CertificateUnknown	`0x2E`
NoCertificate	`0x29`
IllegalParameter	`0x2F`

2.1.3.2 Specifikace typu chyby

Toto pole identifikuje specifickou chybu, která způsobila zaslaní varovné zprávy. Velikost pole je jeden bajt, který odkazuje na jedno z dvanácti specifických čísel, a může obsahovat jednu z možných chyb popsaných v tabulce 2.1. U protokolu TLS byly přidány další chybové kódy, které jsou vypsány v tabulce 2.2.

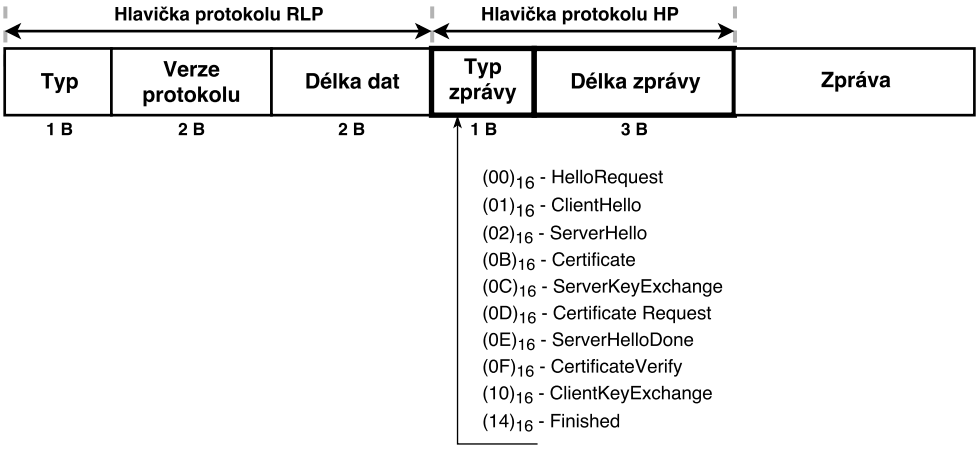
2.1.4 Protokol pro navázání spojení

Posledním podprotokolem protokolu SSL je protokol pro navázání spojení, který je anglicky nazýván jako Handshake Protocol, zkráceně HP. Hlavním úkolem HP protokolu je navázání komunikace a vytvoření bezpečného spojení.

Tabulka 2.2: Přidané hodnoty pro protokol AP v TLS 1.0 [5].

Název chyby	**Kód chyby**
UnknownCA	0x30
ProtocolVersion	0x46
AccessDenied	0x31
InsufficientSecurity	0x47
DecodeError	0x32
InternalError	0x50
DecryptError	0x33
UserCanceled	0x5A
ExportRestriction	0x3C
NoRenegotiation	0x64

Během této komunikace je autentizován server, případně i klient, pokud je jeho autentizace vyžadována. Dále si server s klientem vymění nonce a data potřebná pro výpočet sdíleného tajemství.

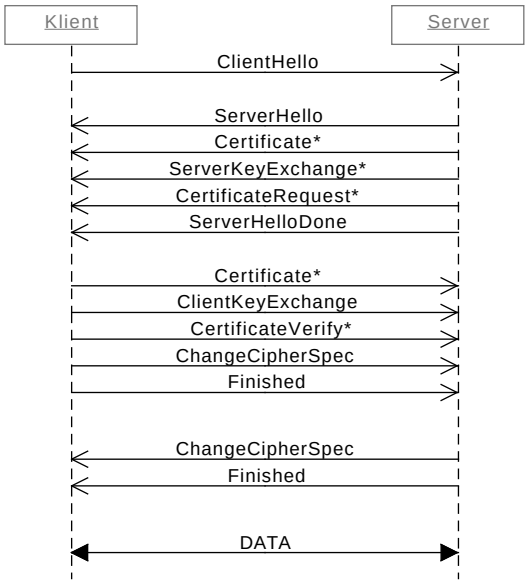


Obrázek 2.5: Formát paketu protokolu HP [4].

Zprávy, ze kterých se navazování spojení skládá, jsou:

- ClientHello
- ServerHello

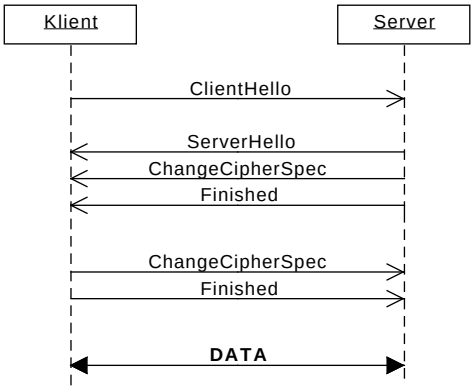
- ServerKeyExchange
- ServerHelloDone
- ClientKeyExchange
- ChangeCipherSpec
- Finished
- ChangeCipherSpec
- Finished



Obrázek 2.6: Tok zasílaných zpráv při úplném navázání spojení[1][1].

Pro navázání spojení jsou možné dvě varianty komunikace. Pokud je vytvářeno úplně nové spojení, probíhá komunikace mezi serverem a klientem, jak je zobrazeno na obrázku 2.6. Když se server s klientem rozhodnou obnovit předchozí relaci nebo duplikovat existující relaci, namísto vyjednávání o nových bezpečnostních parametrech, je komunikace kratší, jak je zobrazeno na obrázku 2.7. Význam jednotlivých zpráv je vysvětlen dále v jednotlivých podsekcích.

[1]Zprávy označené hvězdičkou nemusí být posílány.



Obrázek 2.7: Tok zasílaných zpráv při zkráceném navázání spojení.[1]

2.1.4.1 Zpráva ClientHello

Tato zpráva je posílána jako první, vždy když je navazováno nové spojení se serverem, nebo klient žádá o obnovu stávajícího spojení. Klient může poslat zprávu ClientHello na žádost serveru nebo z vlastní iniciativy s žádostí o znovusjednání bezpečnostních parametrů u existujícího spojení.

Zpráva ClientHello má pevně danou strukturu. Skládá se z polí pro verzi protokolu, struktury s náhodnou hodnotou, identifikačního čísla relace, seznamu žádaných šifrovacích sad a identifikátorů podporovaných kompresních algoritmů.

Identifikátor relace určuje relaci mezi serverem a klientem. Pokud je vytvářeno nové spojení, je tato položka nulová. V případě navazování dřívějšího spojení, současného spojení, nebo jiného aktuálně aktivního spojení, obsahuje první bajt délku identifikace relace. Na základě ID server ví, jaké bezpečnostní parametry si žádá klient znovu použít.

V seznamu žádaných šifer zasílá klient seznam jím podporovaných šifer seřazený podle svých preferencí. Tento seznam je uvozen dvoubajtovou délkou, za kterou následují dvou bajtové řetězce, identifikující šifrovací sadu a algoritmus pro výpočet kontrolního součtu. Server si vybere jednu sadu. Pokud není v seznamu žádná přijatelná volba, je navráceno varování o chybném navázání spojení a spojení je ukončeno.

Poslední položka obsahuje seznam klientem podporovaných kompresních algoritmů seřazené podle jeho preferencí. Seznam je uvozen 1 bajtovou délkou následovaný jednobajtovými identifikacemi jednotlivých kompresních algoritmů. Pokud v seznamu není žádný algoritmus, který by server podporoval, relace je ukončena.

2.1.4.2 Zpráva ServerHello

Po zpracování zprávy ClientHello serverem, odpovídá server tím, že zašle buď varování o chybném navázání spojení, nebo ServerHello zprávu. ServerHello zpráva obsahuje stejná pole jako ClientHello zpráva s tím rozdílem, že obsahuje rozhodnutí serveru, jaká verze protokolu, šifrovací sady, kompresního algoritmu se bude používat. V identifikátoru relace je vrácena stejná hodnota, pokud je ID nalezeno v paměti serveru a server je ochoten navázat nové spojení pomocí zadaného stavu relace. V opačném případě pole identifikátoru obsahuje rozdílnou hodnotu, oproti té zaslané klientem, která identifikuje novou relaci. Pole náhodné struktury obsahuje strukturu vygenerovanou serverem, která je rozdílná a nezávislá na náhodné struktuře zaslané v ClientHello zprávě.

2.1.4.3 Zpráva Certificate

Pokud má být server ověřen, což je žádoucí, odesílá svůj certifikát hned po zaslání ServerHello zprávy. Typ certifikátu musí být vhodný pro zvolenou šifrovací sadou, obsahující algoritmus výměny klíčů. Většinou se jedná o certifikát typu X.509 v3, nebo jinou upravenou verzi certifikátu X.509. Stejný typ zprávy posílá i klient jako odpověď na žádost serveru o certifikát.

Klient tuto zprávu posílá jen tehdy, pokud server poslal zprávu CertificateRequest. Klient pošle serveru svůj certifikát, pokud nemá žádný vhodný certifikát, zasílá místo certifikátu varování, že nemá žádný certifikát (kód `NoCertificate`, vizte tabulka 2.1). Pokud server vyžaduje ověření klienta, reaguje vyvoláním výstrahy selhání navázání spojení.

2.1.4.4 Zpráva ServerKeyExchange

Server posílá zprávu ServerKeyExchange klientovi, pokud nemá žádný certifikát, nebo má certifikát pouze pro podepisování. Tato zpráva se nepoužívá, pokud certifikát od serveru obsahuje Diffie-Hellmanovy parametry.

2.1.4.5 Zpráva CertificateRequest

Neanonymní server může případně od klienta požadovat certifikát, pokud je to vhodné pro zvolenou šifrovací sadu. Žádost je odeslána zprávou CertificateRequest a klient na ni odpovídá zasláním zprávy Certificate.

2.1.4.6 Zpráva ServerHelloDone

Po dokončení komunikace ze strany serveru ke klientovi, pošle server klientovi zprávu ServerHelloDone. Když klient obdrží tuto zprávu, tak ví že server mu už další zprávu nepošle a může nyní vysílat on. V běžném životě by se to dalo přirovnat ke komunikaci přes vysílačku. Ve chvíli kdy jedna strana zahlásí „přepínám“, muže bezpečně mluvit protistrana.

2.1.4.7 Zpráva ClientKeyExchange

Formát zprávy ClientKeyExchange je závislý na volbě algoritmu pro veřejný klíč. U SSL jsou tři možnosti výměny klíče. Buď je to pomocí RSA, FORTEZZA KEA [7] nebo Diffie-Helman.

2.1.4.8 Zpráva CertificateVerify

Tato zpráva se používá k poskytnutí naprostého ověření klientova certifikátu. Zpráva je poslána spolu s jakýmkoliv klientovým certifikátem s podpisovou schopností, což jsou všechny certifikáty kromě těch, které obsahují Diffie-Hellmanovy parametry.

2.1.4.9 Zpráva ChangeCipherSpec

Odeslání zprávy ChangeCipherSpec ze strany serveru nebo klienta znamená, že si jedna nebo druhá strana nastavila přenos z nedůvěryhodného stavu na bezpečný. Ve chvíli, kdy jedna ze stran pošle zprávu ChangeCipherSpec, znamená to, že daná strana změnila bezpečné parametry na své straně spojení na dohodnuté předchozími zprávami. Tato zpráva není součástí zpráv HP protokolu, ale jedná se o zprávu CCSP protokolu popsaného výše.

2.1.4.10 Zpráva Finished

Zpráva Finished je vždy posílána okamžitě za zprávou ChangeCipherSpec, aby bylo ověřeno, že výměna klíčů a autentizace proběhly úspěšně. Tato zpráva je jako první chráněna dohodnutými parametry. Není požadováno potvrzení přijetí zprávy Finished. Strany mohou začít posílat šifrovaná data okamžitě po odeslaní zprávy Finished. Příjemce zprávy ji ještě musí ověřit, že je obsah správný.

2.2 Útoky na SSL/TLS

Tato sekce obsahuje popis útoků na rodinu protokolů SSL a TLS. Budou popsány útoky BEAST, CRIME, POODLE a Lucky13. Jedná se o výběr útoků, které jsou v současné době možné provést na rodinu protoků SSL/TLS. Výběr není rozhodně kompletní a vyčerpávající, ani popisy útoků nezasabíhají do úplných detailů. Záměrem je popsat podstatu každého ze zde zmíněných útoků.

2.2.1 BEAST

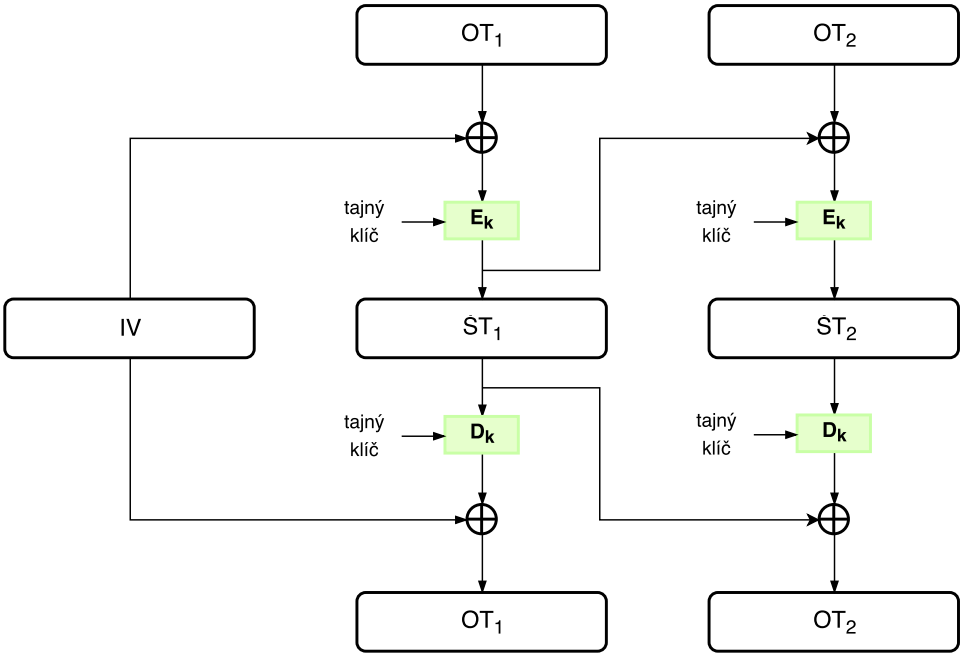
Tato sekce se zabývá útokem BEAST [8, 9]. Jedná se o útok na SSL/TLS používající šifrování s operačním módem CBC s predikovatelným inicializačním vektorem.

Tuto zranitelnost SSL/TLS popsal již v roce 2004 Gregory V. Bard [10]. Avšak až do roku 2011 byla považována pouze za teoretickou. Proveditelnost útoku v praxi byla předvedena na bezpečnostní konferenci Ekoparty v září roku 2011. Útok provedli vědci Thai Doung a Juliano Rizzo proti službě Paypal. Tato služba byla vybrána z důvodu velkého využívání uživateli na internetu a silného zabezpečení.

Chyba má kód CVE-2011-3389, pro kterou Doung a Rizzo také navrhli opravu. Tato oprava byla vytvořena ve spolupráci s hlavními vydavateli webových prohlížečů, jako jsou Mozilla nebo Google. Ačkoliv tato zranitelnost je kryptografického typu, vyžaduje určité podmínky, aby byla úspěšná.

2.2.1.1 Popis zranitelnosti

Standard SSL umožňuje při použití symetrického klíče, šifrovat buď pomocí blokových nebo proudových šifer. Většina implementací používá blokové šifry, proto se v popisu zranitelnosti zaměříme na situaci, kdy je použita bloková šifra. Bloková šifra pracuje s daty pevně dané délky, zvané bloky. Například šifra AES [11] pracuje nad bloky velikosti 128 bitů.



Obrázek 2.8: Diagram šifrování a dešifrování s použitím operačního módu CBC [12].

Při šifrování dat delších než velikost bloku, jsou použity operační módy šifrování. SSL používá jako hlavní operační mód CBC [12], který zprávu $OT = OT_1, \ldots, OT_n$, kde délka každého bloku OT_i $(1 \leq i \leq n)$ je ma-

ximalní délka bloku podporovaná danou šifrou, zašifruje následovně: vezme inicializační vektor, který označme IV a vypočte $\check{S}T_1, \ldots, \check{S}T_n$ jako:

$$\check{S}T_1 = E_k\left(OT_1 \oplus IV\right)$$

$$\check{S}T_i = E_k\left(OT_i \oplus \check{S}T_{i-1}\right).$$

$E_k(X)$ značí funkci šifrování nějakého bloku X pomocí klíče k. V horní části obrázku 2.8 je znázorněno šifrování bloků pomocí operačního módu CBC.

Výsledný řetězec bloků šifrového textu $\check{S}T_1, \ldots, \check{S}T_n$ je poslán protistraně. Součástí šifrové zprávy je i blok IV, pokud ho již protistrana nezná. Po přijetí si příjemce zprávu dešifruje, aby získal původní zprávu OT, pomocí následujícího vztahu:

$$OT_1 = D_k(\check{S}T_1) \oplus IV$$

$$OT_i = D_k(\check{S}T_i) \oplus \check{S}T_{i-1}.$$

Dešifrování bloku X pomocí klíče k je ve vzorci označeno jako $D_k(X)$ a jedná se o inverzní funkci k šifrovací funkci. V dolní části obrázku 2.8 je znázorněno dešifrování bloků pomocí operačního módu CBC.

Inicializační vektor je blok náhodných dat, který slouží k úpravě prvního datového bloku otevřeného textu. V protokolu SSL je inicializační vektor, pro šifrování první zprávy, vytvořen z nonce serveru a klienta, získané při navazování spojení. Pro všechny další zprávy je jako inicializační vektor použit poslední šifrový blok předcházející zprávy. Z tohoto důvodu není takto tvořený inicializační vektor bezpečný, jelikož ho útočník dokáže odposlechnout. Bezpečnější by bylo pokaždé volit inicializační vektor náhodně na základě hodnot, které útočník nezná a neměl by tak možnost IV odhadnout, ani odposlechnout, jako je tomu u IV pro první zprávu.

Útočník na základě odposlechnuté šifrované komunikace dokáže vkládat do komunikace svůj vlastní otevřený text a na základě porovnání šifrových bloků určit tajemství obsažené v komunikaci mezi klientem a serverem. Předpokládejme, že útočník dokáže odposlechnout několik za sebou jdoucích šifrových bloků $\check{S}T_1, \ldots, \check{S}T_n$ i s inicializačním vektorem a chce zjistit tajemství OT^*, které je obsaženo v bloku OT_j, pro který platí že $1 \leq j \leq n$, otevřeného textu. Vzhledem k odposlechu a znalosti konstrukce šifrování SSL protokolu za využití CBC módu zná útočník IV, který je použit pro šifrování následné zprávy. Jedná se o poslední šifrový blok $\check{S}T_l$ předchozí zprávy.

Díky těmto znalostem může útočník zkonstruovat zprávu OT'. Tato zpráva má počáteční blok OT'_1 zkounstruován jako $\check{S}T_{j-1} \oplus \check{S}T_l \oplus OT^*$, kde $\check{S}T_{j-1}$ je šifrový blok se kterým byl exkluzivně sečten blok OT_j (vizte obr. 2.8), pro který chce útočník určit hodnotu tajemství OT^*. Nyní když útočník přinutí klienta, aby zašifroval jeho zprávu OT', stane se následující:

$$\check{S}T'_1 = E_k(OT'_1 \oplus \check{S}T_l) = E_k(\check{S}T_{j-1} \oplus \check{S}T_l \oplus OT^* \oplus \check{S}T_l) = E_k(\check{S}T_{j-1} \oplus OT^*).$$

Takto dokáže útočník uhodnout hodnotu tajemství obsaženou v bloku OT_j. Díky odposlechu zná útočník hodnotu $\check{S}T_j = E_k(OT_j \oplus \check{S}T_{j-1})$. Z této znalosti lze odvodit, že bloky $\check{S}T'_1$ a $\check{S}T_j$ se rovnají, jen tehdy když se rovnají bloky OT_j a OT^*. Opakovaným zkoušením různých hodnot bloku OT^* lze zjistit hodnotu jakéhokoliv bloku OT_j otevřeného textu.

2.2.1.2 Požadavky na útok

Aby byl útok proveditelný, je třeba splnění určitých požadavků. Při klasické komunikaci mezi serverem a klientem se jedná o tři aspekty, které jsou dále popsány.

Prvním aspektem je schopnost útočníka pasivního síťového odposlechu. Útočník musí být schopný zachytávat šifrované HTTPS dotazy, které jsou posílány klientem. Toto mu umožní zachytávat šifrové texty jak od serveru tak od klienta. Dokaže tak zachytit i IV pro jakýkoliv daný šifrový blok.

Dalším požadavkem je, aby měl útočník oprávnění k určování pozice tajemství ve zprávě vzhledem k hranicím bloků. U běžné webové komunikace může útočník ovlivnit pozici tajemství ve zprávě pomocí přidání URL parametrů, nebo změnou hlavičky. Úpravou hranice šifrových bloků, může útočník zkoušet a vytvářet bloky, které jsou snadněji odhadnutelné. Takto lze vytvořit blok obsahující pouze jeden neznámý bajt, který lze snadněji uhodnout.

Posledním poožadavkem je oprávnění pro vložení zvoleného otevřeného textu. Útočník dokáže odhadnout hodnotu otevřeného textu spojenou s šifrovým textem zachyceným pasivním síťovým odposlechem. Konkrétně útočník dokáže vložit svůj vybraný otevřený text do otevřeného textu, který je šifrován. Z pravidla to znamená, že útočník může použít prohlížeč klienta jako šifrovací orákulum. Kromě toho je třeba udržovat stejnou SSL/TLS relaci napříč více různých vložení zvolených otevřených textů. [9]

2.2.1.3 Shrnutí

Pokud útočník dokáže odposlouchávat komunikaci mezi klientem a serverem a ovládnou prohlížeč klienta, ještě stále nemá jistotu, že provede úspěšný útok BEAST. Jak bylo popsáno výše, útok BEAST využívá chyby v implementaci protokolu SSL 3.0 a TLS 1.0, který navíc podporují šifrové sady, které používají při šifrování operační mód CBC. Možnost navázat komunikaci se server s využitím protokolu SSL 3.0 nebo TLS 1.0 a šifrové sady, která využívá při šifrování operační mód CBC, je pro útočníka neméně důležitá. Z tohoto důvodu lze tvrdit, že server je zranitelný vůči útoku BEAST, pokud umožňuje navázat komunikaci s klientem za využití protokolu SSL 3.0 nebo TLS 1.0 a šifrové sady využívající operační mód CBC.

V současné době jsou opraveny chyby v prohlížečích nebo dalších aplikacích, které využívá BEAST při útoku. Nicméně podpora zranitelných šifrových sad a protokolů ze strany serverů zůstává. Podle statistik SSL Pulse [13] je ak-

tuálně 91,6% serverů zranitelných vůči útoku BEAST. Pro reálnější představu se jedná o 129 313 serverů, které jsou zranitelné. Jedná se až o neskutečné číslo, když vezmeme v potaz, že je tento útok znám od roku 2011.

2.2.2 CRIME

Další útok popsaný v této sekci je CRIME [14, 9, 15, 16]. Jedná se o zkratku pro útok postranními kanály na protokol SSL/TLS. Písmeno „C“ v názvu útoku znamená kompresi (angl. compression). Komprese slouží ke snížení počtu bitů stejného množství dat, která jsou přenášena nebo uložena na internetu. „R“ značí anglické ratio, což znamená poměr. Představuje poměr nadbytečnosti ve zprávě, čím vetší nadbytečnost tím je lepší komprese. Písmeno „I“ je od anglického information leak, což znamená únik informace. Tento únik je způsoben, protože protokol SSL/TLS nedokáže skrýt délku dotazu na server, ani odpovědi serveru. Posledními písmeny v názvu jsou „ME“, které jsou zkratkou pro anglické mass exploitation nebo made easy. Českým ekvivalentem pro první anglický název je hromadně zneužitelný. Tento přívlastek byl použit, protože v době vzniku útoku bylo možné napadnout 45% prohlížečů, všechny servery používající SPDY [17], což je volný síťový protokol vyvinut především společností Google, který slouží pro přenos webového obsahu, (např.: Twitter, Gmail) a 40% SSL/TLS serverů (např.: Dropbox, GitHub). Druhý význam zkratky „ME“ v češtině znamená snadno proveditelný.

Zranitelnost v kompresních metodách je popsána od roku 2002, kdy ji popsal kryptolog John Kelsey[18]. Útok CRIME (CVE-2012-4929) využívající tuto zranitelnost, byl demostrován vědci Juliano Rizzo a Thai Duong na bezpečnostní konferenci Ekoparty v roce 2012. Vědci demonstrovali útok na produktech velkých programových vývojářů jako jsou Mozilla, Google nebo Dropbox. Tento útok funguje nejen proti kompresi SSL/TLS nebo SPDY, ale zranitelné jsou také další šifrovací a kompresní protokoly.

2.2.2.1 Popis zranitelnosti

Protokol SSL/TLS obsahuje bezztrátovou kompresní metodu, která umožňuje přenesení stejného množství dat v méně bitech. Tato bezztrátová kompresní metoda se nazývá DEFLATE [19], který se skládá ze dvou podalgoritmů: Lempel-Ziv kódování, neboli LZ77, a Huffmanovo kódování. LZ77 slouží k eliminaci nadbytečnosti opakujících se sekvencí, zatímco Huffmanovo kódování slouží k eliminaci nadbytečnosti opakujících se symbolů. DEFLATE skenuje vstup a hledá opakující se řetězce, které nahrazuje zpětným odkazem na poslední výskyt. Hlavní zajímavostí této kompresní techniky je velikost okénka, které má hodnotu mezi 1 a 15. Platí, že čím vyšší velikost okénka, tím vyšší kompresní poměr. Algoritmus má omezení na vzdálenost od posledního výskytu (32 tisíc bajtů) a na délku sekvence (258 bajtů).

Jestli bude použita komprese a případně jaký kompresní algoritmus bude použit, je dohodnuto během navazování spojení SSL/TLS protokolem. Ve zprávě ClientHello zasílá klient seznam podporovaných kompresních algoritmů. Server odpovídá, pomocí zprávy ServerHello, jaký si z nabízených kompresních algoritmů vybral. Pokud je vybrán nějaký kompresní algoritmus, je komprese použita na všechna aplikační data, která jsou posílána protistraně. Například při posílání dat HTTP je komprimována i hlavička HTTP dotazu nebo odpovědi.

Velikost obsahu je dána `délka(šifrování(komprese(data)))`. Z tohoto postupu je možné vidět, že i když dojde k zašifrování obsahu zprávy, naskryje se tím délka komprimované zprávy. Útočník také ví, že součástí HTTP dotazu od klienta je `Cookie: secretcookie=`, které obsahuje tajnou hodnotu, kterou chce utočník získat.

Útočník odhaduje hodnotu skrytou v `secretcookie` pomocí hrubé síly. V prvním pokusu vloží útočník do hlavičky HTTP dotazu `secretcookie=0` a nechá provést kompresi. V případě, že je útočníkem vložený text v HTTP dotazu již obsažen, je délka zprávy po kompresi menší než když vložený text v HTTP dotazu není. Tato vlastnost je jádrem celého útoku. Útočník postupně vkládá různé odhady, a následně měří a porovnává délky odhadů tajemství.

Kompresní funkce během svého běhu nahradí shodnou sekvenci malou značkou. V našem případě je nahrazena sekvence `secretcookie=`, která bude nahrazena značkou určující délku 13 znaků a pozici `n` bajtů zpět. Předpokládejme, že odhadovaný znak tajné hodnoty, například znak `0`, není spravným tipem, pak kompresní funkce pro tento znak vytvoří extra značku. Takto postupně útočník zkouší jednotlivé znaky, kde budou mít všechny stejnou kompresní délků až na jediný. Tím je pokus s platnou hodnotou hádaného tajemství. Tento pokus bude mít lepší kompresi, z čehož plyne, že bude mít o jeden bajt kratší obsah zprávy. Nyní se může útočník posunout o jeden bajt dále a celý proces zopakovat pro další znak, dokud nezíská celé tajemství.

2.2.2.2 Požadavky na útok

Pro úspěšný útok musí být splněny určité požadavky. Jako první musí být utočník schopný odposlouchávat komunikaci mezi klientem a serverem. Dalším požadavkem je, že se oběť autentizovala pomocí HTTPS a má sjednané SSL/TLS spojení se serverem. Nakonec musí útočník přinutit oběť ke vstupu na své webové stránky, na kterých běží jeho útočný skript.

Pro úspěšný útok musí utočník dostat svůj útočný kód do prohlížeče oběti. Toho docílí například tím, že obět vstoupí na jeho webové stránky, na kterých je tento útočný kód uložen. Jakmile je prohlížeč oběti získán, přinutí útočník tento prohlížeč opakovaně se dotazovat na HTTPS webové stránky.

2.2.2.3 Shrnutí

Ačkoliv se jedná v první řadě o útok na samotného klienta, i server musí splnit požadavky, aby byl útok možný. Z výše zmíněného popisu zranitelnosti, je zřejmé, že problém je v použití komprese pro zrychlení komunikace. Tento útok, na rozdíl od útoku BEAST, není závislý na verzi protokolu SSL nebo TLS ani na použitém šifrování. Pokud server podporuje kompresi dat, potom je vždy zranitelný.

Oproti roku 2012, kdy byl útok CRIME představen, se zranitelnost serverů vůči tomuto útoku velice snížila. V současné době podle statistik SSL Pulse [13] používají SSL/TLS kompresi 3% serverů a SPDY 6,1% serverů. Což nejsou tak hrozivá čísla jako při zrodu útoku CRIME. Přesto se stále jedná o 8596 serverů, které podporují SPDY, a 4238 serverů, které podporují SSL/TLS kompresi.

2.2.3 POODLE

V roce 2014 se objevil další útok na protokol SSL 3.0. Jedná se o útok zvaný POODLE [20, 21, 22], který využívá zranitelnosti v protokolu SSL 3.0. Možnost tohoto útoku odhalili bezpečnostní technici společnosti Google Bodo Möller, Thai Duong a Krzysztof Kotowicz.

I když server i klient podporují nejnovější protokoly TLS, vytvoření spojení za použití SSL 3.0 je stále možné. Je to hlavně kvůli vyhnutí se chyb ve vnitřní operabilitě serverů. Z tohoto důvodu jsou zejména na straně klientů implementovány mechanismy, které umožňují v případě potřeby degradovat protokol spojení. Útok POODLE využívá právě této možnosti degradování spojení na komunikaci pomocí protokolu SSL 3.0, ačkoliv by byla možná komunikace pomocí protokolů TLS.

Pokud není zakázání používání protokolu SSL 3.0 zcela žádoucí, měla by implementace TLS využívat `TLS_FALLBACK_SCSV` [23]. Jedná se o speciální šifrovou sadu, kterou by měl klient zasílat ve zprávě ClientHello vždy, když chce komunikovat se serverem s nižší verzí protokolu SSL/TLS než je jeho nejvyšší podporovaná verze. Pokud server obdrží v seznamu šifrových sad `TLS_FALLBACK_SCSV`, zkontroluje jakou verzí protokolu chce klient komunikovat. V případě, že server podporuje vyšší verzi protokolu SSL/TLS než je verze obsažená ve zprávě ClientHello, potom server pošle klientovi chybu `inappropriate_fallback`. Pokud je verze obsažená ve zprávě ClientHello nejvyšší verzí, kterou server podporuje, nebo v seznamu šifrových sad není `TLS_FALLBACK_SCSV`, potom server naváže komunikaci s klientem jak je obvyklé. Tato šifrová sada tedy slouží k obraně proti navazování spojení za použití zbytečně nízké verze protokolu SSL/TLS.

2.2.3.1 Popis zranitelnosti

Základem celé zranitelnosti je umožnění použití protokolu SSL verze 3.0, který je v současné době označen jako nedostatečně bezpečný. Tento protokol používá pro šifrování zpráv buď proudové, nebo blokové šifry. Na tento útok jsou zranitelné blokové šifry, které využívají pro šifrování a dešifrování operační mód CBC. Další slabinou, kterou POODLE využívá, je fakt, že protokol SSL 3.0, na rozdíl od protokolu TLS a novějších, vynechává kontrolu určitých částí dat obsažených v každé zprávě. Těchto zranitelností dokáže útočník využít k dešifrování jednotlivých slabik zpráv.

Nejvážnějším problémem u šifrovaní s operačním módem CBC v protokolu SSL 3.0 je, že výplň bloků šifry není deterministická a není zahrnuta do výpočtu MAC. Proto se nedá plně ověřit integrita výplně po dešifrování. Výplň slouží k doplnění bloku na plnou velikost. Výplň je dlouhá 1 až L bajtů, kde L je maximální velikost bloku zvolené šifry. Nejsnadněji se dá využít slabosti ve výplni, když je celý blok vyplněn $L-1$ libovolnými bajty a poslední bajt obsahuje hodnotu $L-1$. Z obdrženého šifrového textu $\check{S}T_1, \ldots, \check{S}T_n$ a $\check{S}T_0 = IV$ je dešifrováním, podle vztahu $OT_i = D_k(\check{S}T_i) \oplus \check{S}T_{i-1}$, kde D_k je dešifrování podle zvoleného klíče, získán otevřený text $OT_1, \ldots, OT_n$. Následuje kontrola a odstranění výplně z konce zprávy, a kontrola a odstanění kontrolního součtu MAC. Za předpokladu, že poslední blok je jen vycpávka, a útočník provede náhradu posledního bloku $\check{S}T_n$ za jiný blok $\check{S}T_i$, který se nachází někde dříve ve stejném šifrovém textu, bude celý šifrový text přijat pouze tehdy, když dešifrování $D_k(\check{S}T_i) \oplus \check{S}T_{n-1}$ dá blok, který bude v poslední slabice mít hodnotu $L-1$. V ostatních případech bude se vší pravděpodobností odmítnut. Toto je princip útoků na výplně (anglicky „padding oracle attack“) [24].

Předpokládejme, že útočník má kontrolu nad cestou dotazu i tělem dotazu, stejně tak zná délku hledané hodnoty parametru „cookie“. Ze znalosti použité šifry a protokolu, pak zná velikost šifrových bloků a délku MAC, která je typicky u SSL 3.0 20 bajtů. Vzhledem k těmto znalostem musí útočník vytvořit dotaz tak, aby platilo, že výcpávka zabírá celý jeden blok, po zašifrování se jedná o blok $\check{S}T_n$, a první neznámý znak hodnoty parametru „cookie“ je poslední bajt nějakého předešlého bloku, po zašifrování se jedná o blok $\check{S}T_i$. Nyní provede útočník náhradu bloku $\check{S}T_n$ blokem $\check{S}T_i$ a pošle takto upravený SSL záznam serveru. Ve většině případů dojde k chybě a ukončení spojení. V takovém případě to jednoduše zkusí útočník znovu. Avšak průměrně v jednom z 256 dotazů server zprávu příjme. V takovém případě platí, že $D_k(\check{S}T_i)[L] \oplus \check{S}T_{n-1}[L] = L-1$. Z této informace dokáže útočník zrekonstruovat jeden bajt hodnoty parametru „cookie“, pomocí vztahu $OT_i[L] = L-1 \oplus \check{S}T_{i-1}[L] \oplus \check{S}T_{n-1}[L]$. Tento postup se opakuje dokud útočník nezíská celou hledanou informaci.

2.2.3.2 Požadavky na útok

Pro samotný útok je potřeba určitých prerekvizit. Zaprvé útočník musí být schopný zachytávat komunikaci klienta, například pomocí veřejné bezdrátové sítě. Dalším úkolem, který musí útočník provést, je vytvoření skriptu, který přinutí prohlížeč oběti posílat HTTPS dotazy na danou adresu, zachytávat a upravovat SSL záznamy tak, aby odhadl tajnou informaci.

2.2.3.3 Shrnutí

Ačkoliv je pro útok potřeba infiltrovat prohlížeč oběti, musí i server splňovat některé podmínky. Útok POODLE využívá zranitelnosti v protokolu SSL verze 3.0. Z tohoto důvodu, pokud server podporuje protokol SSL 3.0 a šifrové sady s operačním módem CBC, je zranitelný.

V současné době se dá považovat počet zranitelných serverů za nizký. Podle statistik SSL Pulse [13] je 2,9% serverů zranitelných na útok POODLE, a u dalších 1,3% serverů není toto riziko známo, což nám dohromady dává teoretický počet 5834 zranitelných serverů na útok POODLE. Navíc i nové výrobky, které spadají do kategorie internetu věcí, mohou být zranitelné tímto známým typem útoku, například panenka Hello Barbie [25].

2.2.4 LUCKY 13

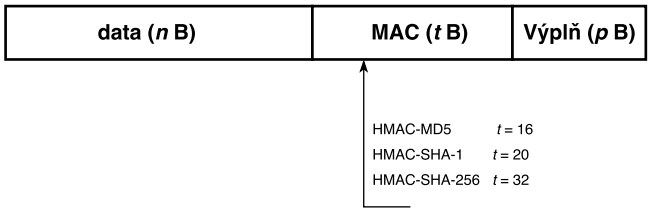
Posledním útokem, který cílí na protokol SSL/TLS, zmíněným v této práci je „Lucky Thirteen“ [26, 27, 28, 9]. Tento anglický název útoku se dá do češtiny přeložit jako „štastných třináct“. Název útoku je ukázkou humoru mezi kryptografy, jelikož číslo 13 je obecně považováno za neštastné číslo, nicméně zde je právě díky 13 bajtům tento útok možný. Proto byl útok pojmenován „Lucky 13“.

„Lucky 13“ je časový útok postranními kanály, který představili Nadhem AlFardan a Kenny Paterson v roce 2013. Tento útok umožňuje útočníkovi, který odposlouchává komunikaci šifrovanou pomocí CBC módu, zrekonstruovat původní otevřený text. Útok je možné provést na TLS nebo DTLS protokoly. DTLS je protokol pro zabezpečení komunikace pomocí datagramů jako je například spojení pomocí UDP protokolu. „Lucky 13“ využívá stejného mechanismu jako útok na výplně (anglicky „padding oracle attack“) [24].

2.2.4.1 Popis zranitelnosti

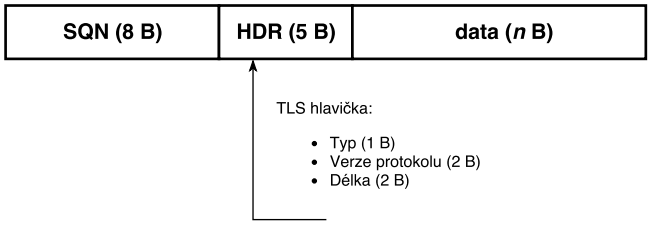
Celá přenášená šifrová zpráva, která je složena z několika bloků o velikosti dané použitou blokovou šifrou, se skládá ze tří částí (vizte obr. 2.9). V první části jsou data, následována značkou pro kontrolní součet MAC. MAC je vypočítána ze zprávy složené ze tří částí, vizte obr. 2.10. Jedná se o sekvenci, která se s každým odeslaným záznamem zvětšuje, hlavičku TLS RLP protokolu a samotná aplikační data. V závislosti na použitém algoritmu pro výpočet MAC

je dána velikost MAC značky. Poslední částí je výplň, která doplňuje celá data na velikost násobku velikosti bloku zvolené blokové šifry.



Obrázek 2.9: Struktura zprávy, která je zašifrována pro přenos mezi klientem a serverem.

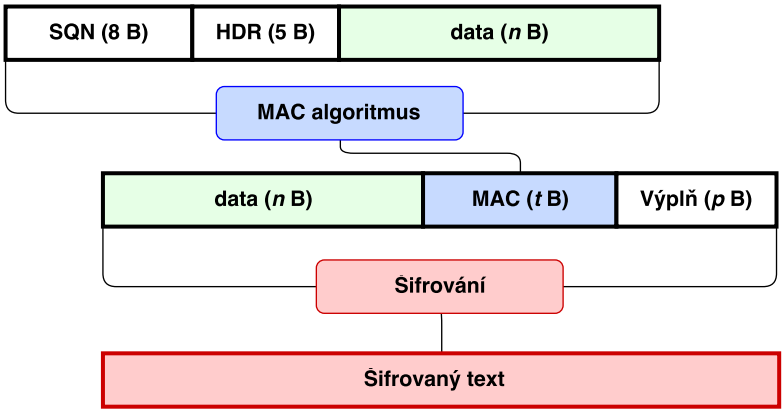
Server po přijetí šifrového bloku provede tři operace. Nejprve si dešifruje přijatou zprávu. Následně ověří hodnotu posledního bajtu v posledním bloku, která obsahuje počet bajtů výplně. Pokud se jedná o platnou hodnotu je výplň odebrána, v opačném případě server vyvolá chybu výplně. Po úspěšném odebrání výplně je z dat vypočítán kontrolní součet podle dohodnutého algoritmu a zkontrolován se zaslanou hodnotou MAC. V případě shody jsou data přijata a považována za důvěryhodná a ověřená, jinak je serverem vyvolána chyba kontrolního součtu. Jelikož výplň není součástí kontrolního součtu, nedá se ověřit její správnost. Této znalosti využívá útok na výplně, který je základem pro „Lucky 13“.



Obrázek 2.10: Struktura dat, předaných MAC algoritmu pro výpočet kontrolního součtu.

Nové verze protokolu TLS se snaží zabránit tomuto typu úniku informace. Současné verze protokolu TLS 1.1 a vyšší se snaží branit úniku informace tak, že server ukončí relaci v případě chyby výplně nebo MAC, což je rozdíl oproti TLS 1.0 a SSL 3.0. Tímto způsobem je útočníkovi zabráněno v opakovaném posílání požadavků na dešifrování zašifrované zprávy. Nicméně útok na výplň je možný i pomocí více různých relací, pokud oběť útoku obnoví relaci a tajemství se nachází na stejné pozici v každé zprávě. Protože relace je ukončena dříve, pokud došlo k chybě výplně než, když dojde k chybě MAC. Implementace TLS 1.2 se snaží zabránit těmto časovým únikům, tím že provede výpočet MAC, i když dojde k chybě výplně. Pokud dojde k chybě výplně

je kontrolní součet počítán, jako by nebyla ve zprávě obsažena žádná výplň. Z čehož plyne, že MAC bude spočítána vždy špatně, navíc pokud je porušena výplň výpočet MAC může trvat o něco déle. Právě této časové chyby využívá útok „Lucky 13“.



Obrázek 2.11: Postup zpracování dat až po zašifrování při použití blokové šifry s CBC módem a HMAC algoritmem.[26]

Předpokládejme, že máme již upravenou šifrovou zprávu $\check{S}T^*$, kterou server dešifruje na otevřený text OT^*, který se skládá ze čtyř bloků $OT_1||OT_2||OT_3||OT_4$. Pro šifrování a dešifrování byla dohodnuta šifra AES, která má velikost bloku 16 bajtů a pro výpočet MAC je použit algoritmus HMAC-SHA-1, který vytváří MAC značku délky 20 bajtů. Po dešifrování mohou nastat pouze tři situace:

1. Poslední bajt v bloku OT_4 je `0x00`. V takovém případě je za výplň považován tento jeden bajt a je odstraněn. 20 bajtů předcházející výplni je MAC a zbylých $64-20-1 = 43$ bajtů je otevřený text, který si označme jako R. Nyní může být poskládána zpráva pro ověření MAC, která se skládá z $SQN||HDR||R$ a je tedy dlouhá $8+5+43 = 56$ bajtů.

2. Blok OT_4 končí platným označením výplně dlouhé nejméně dva bajty. V tomto případě jsou nejméně dva bajty odstraněny jako výplň a dalších 20 bajtů je MAC. Z toho plyne, že otevřený text R je nejvýše 42 bajtů dlouhý a zpráva pro ověření MAC má délku nejvýše 55 bajtů.

3. Blok OT_4 končí neplatným bajtem pro výplň. V takovém případě je s otevřeným textem nakládáno jako by neobsahoval žádnou výplň, takže posledních 20 bajtů je považováno za MAC. Zbylých 44 bajtů je otevřený text R. Podle dříve zmíněné rovnice vypočteme, že délka zprávy pro kontrolu MAC bude dlouhá 57 bajtů.

Ve všech třech výše zmíněných případech selže ověření MAC. Nicméně v prvním a třetím případě je doba výpočtu MAC delší než v druhém případě.

Na základě těchto časových rozdílů dokáže útočník určit, kdy došlo k druhému případu. Tento případ dává útočníkovi možnost získat dva bajty otevřeného textu.

2.2.4.2 Požadavky na útok

Jako u všech dříve zmíněných útoků, i tento vyžaduje odposlech komunikace na síti. Díky odposlechu získá útočník šifrovaná data, která musí upravit podle své potřeby, aby mohl získat tajný obsah zpráv. Za pomoci škodlivého programu na klientově počítači, nebo skriptu na stránkách, které klient navšívil, vytvořit mnoho spojení na server, na který bude útočný kód zasílat požadavky na dešifrování odposlechnuté, upravené šifrované zprávy.

Pro získání jednoho bajtu je zapotřebí tisíce spojení a navázání spojení je časově náročné. Z tohoto důvodu může útočníkovi trvat dny získat tajný obsah. Pro zlepšení útoku je možné využít techniky použité v útoku BEAST.

2.2.4.3 Shrnutí

Ačkoliv základem zranitelnosti na útok „Lucky 13“ je špatná implementace SSL/TLS a DTLS, není to jediný požadavek. Server musí podporovat šifrovou sadu, která využívá při šifrování operační mód CBC a pro výpočet kontrolního součtu používá algoritmus HMAC-MD5, HMAC-SHA-1, nebo HMAC-SHA-256. Pokud server podporuje navázání spojení s takovouto konfigurací, můžeme ho považovat za zranitelný, ačkoliv to nemusí být pravda, pokud bude provedena bezpečná implementace protokolu TLS nebo DTLS.

2.3 Další typy útoků na server

Následující sekce je věnována útokům, které využívají zranitelností v programovém vybavení serverů. Jako zastupce těchto utoků je níže popsán útok Shellshock. Nejedná se však o jediný útok tohoto typu, který se na internetu vyskytuje, nicméně jedná se o nejaktuálnější útok tohoto typu.

2.3.1 Shellshock

Shellshock [29, 30, 31, 32],v angličtině také označován jako Bashdoor, je útok, který využívá chyby přímo v příkazovém procesoru Bash, který je základem operačních systémů UNIX, Linux, ale i Apple a Android.

Bash [33], zkratka pro anglické Bourne-Again shell, je jedním z unixových příkazových procesorů, které interpretují příkazovou řádku. Tento procesor byl vyvynut panem Foxem v roce 1987. Nicméně chyba Shellshock do příkazového procesoru byla nejspíš zavedena až roku 1992. Nedá se to však určit přesně jelikož pro dřívější verze nejsou k dispozici detailní záznamy.

Chybu v „Bash“ jako první zdokumentoval Stéphane Chazelas. Ten v roce 2014 kontaktoval hlavního vývojáře „Bash“, aby ho informoval o nalezení chyby. Tato chyba byla označena CVE kódem CVE-2014-6271, a svou povahou je přirovnávána k chybě Heartbleed [34].

Nicméně Shellshock je mnohem větší hrozba než Heartbleed. Zatímco u chyby Heathbleed šlo, díky špatné implementaci, získat hesla k přístupu na server, chybu Shellshock lze použít k získání celého serveru. Navíc během dvou let, během nichž se o chybě Heartbleed nevědělo, bylo napadeno pět set tisíc serverů. O chybě Shellshock se však nevědělo 22 let.

Objevením této první chyby v příkazovém procesoru vedlo k jeho důkladnému prověření. Během této bezpečnostní kontroly byly objeveny i další chyby podobného typu jako je chyba Shellshock. Všechny byly označeny CVE kódy (CVE-2014-6277, CVE-2014-6278, CVE-2014-7169, CVE-2014-7186, CVE-2014-7187) a byly pro ně zavedeny opravy v „Bash“.

2.3.1.1 Popis zranitelnosti

Shellshock je typem chyby spadající do kategorie chyb spuštění libovolného kódu (ACE) [35]. Útoky tohoto typu bývají velmi obtížné, jelikož je útočný program spuštěn v rámci jiného programu. Aby bylo toto možné, je třeba, aby měl útočník znalosti vnitřního kódu spuštěného programu, formát paměti i jazyka symbolických adres. Avšak zranitelnost Shellshock je velký problém, protože odstraňuje potřebu těchto znalostí a umožňuje snadný, možná až velmi snadný, způsob převzetí kontroly nad jiným počítačem nebo přístrojem.

Při komunikaci serveru s klientem jsou tři místa v HTTP dotazu, kde je možné využít zranitelnosti Shellshock. Těmi místy jsou URL dotaz, hlavička, která je posílána spolu s URL, a posledním místem je pozice pro argumenty. Argumenty obsahují například uživatelské jméno, pokud je zasíláno na server.

Útočník může využít jedno z těchto míst k vložení příkazu pro „Bash“. Nejprve je třeba vložit „magickou“ definici funkce `() { :;};`. Princip je takový, že „Bash“ čte hlavičku HTTP dotazu a narazí na kulaté závorky. V tuto chvíli předpokládá, že je definována funkce, které má tělo ve složených závorkách. Tělo funkce musí obsahovat nějaký příkaz a v „Bash“ platí, že znak `:` je nejmenší příkaz, který lze bez chyby zavolat. Takto je korektně vytvořena funkce a za touto definicí následují příkazy útočníka, které chce po serveru provést.

2.3.1.2 Použití útoku

Shellshock lze použít k velké škále různý útoků. Lze provést sledovací útoky, útoky pro nalákání uživatele na utočníkův web, zařadit server do sítě robotích serverů, které mohou být součástí DOS útoku, a v neposlední řadě lze získat hesla uložená na zranitelném serveru. Jelikož je „Bash“ základem různých operačních systémů, lze provést útok nejen na server, ale i proti jiným síťovým

prvkům, které umožnují spustit CGI skripty. K zabránění celé této škály útoků je potřeba aktualizovat operační systém serveru nebo jiného síťového prvku, jehož součástí je zranitelný příkazový procesor.

2.4 Programy na testování zranitelností

V současné době existuje spousta programů na testování nejrůznějších zranitelností serverů. Značný počet jich existuje jako online testovací služby, kde stačí zadat název domény serveru, nebo IP adresu serveru a webové stránky provedou test serveru na nejaktuálnější zranitelnosti, které v současné době útočníci využívají. Správce serveru si může snadno a rychle otestovat zranitelnost svého serveru, například s použitím COMODO SSL Analyzer[2], FairSSL[3], Pentest-tools[4], nebo SSL Tester[5].

Kromě těchto webových stránek na testování serverů, existují i programy pro operační systémy, které otestují uživatelem zadaný server. Dále je více přiblížen program SSLyze, který je později v práci využiván při implementaci jednotlivých testovacích modulů. Mimo tento program existují například ještě programy Nessus[6], nebo Burp suite[7]

2.4.1 SSLyze

SSLyze je program na testování serverů, které využívají pro komunikaci s klientem SSL/TLS protokol. Program byl vytvořen společností iSEC Partners a je volně k dispozici na verzovacím serveru GitHub[8]. Implementace je provedena v jazyce Python, proto je možné ho spustit, jak na operačních systémech UNIX/Linux, tak na operačních systémech Microsoft Windows.

Program je rozdělen na části, tak aby byl snadněji rozšiřitelný. Testy, které může uživatel využít pro testování serveru, jsou odděleny od hlavní části programu a jsou nahrávány do programu jako zásuvné moduly. K nahrávání zásuvných modulů slouží třída `PluginsFinder`, která provede statické nebo dynamické nahrání zásuvných modulů. Řídídí část celého programu je obsažena v souboru `sslyze_cli.py`. V tomto souboru je funkce `main`, která provede následující body:

- Pomocí třídy `PluginsFinder` najde a načte všechny zásuvné moduly
- Pomocí třídy `CommandLineParser` vytvoří seznam možných přepínačů pro příkazovou řádku

[2]https://sslanalyzer.comodoca.com/
[3]https://www.fairssl.se/en/ssltest
[4]https://pentest-tools.com/
[5]https://www.ssllabs.com/ssltest/index.html
[6]http://www.tenable.com/products/nessus-vulnerability-scanner
[7]https://portswigger.net/burp/
[8]https://github.com/iSECPartners/sslyze

- Analyzuje příkazovou řádku
- Provede test spojení se zadanými servery pomocí třídy `ServerConnectivityTestrer`
- Spustí požadované kontroly na dostupné servery
- Vypíše výsledky testů na příkazovou řádku, anebo do XML souboru

Všechny zásuvné moduly dědí od abstraktní třídy `PluginBase`. Třídy dědící od abstraktní třídy obsahují instanci třídy `PluginInterface`, které slouží pro identifikaci zásuvného modulu pro SSLyze. Třída `PluginInterface` definuje titulek zásuvného modulu, jeho popis, příkaz pro zavolání daného zásuvného modulu a další příkazy spojené se zásuvným modulem. Součástí abstraktní třídy `PluginBase` je i abstraktní metoda `process_task`, která slouží k provedení samotného otestování serveru. Tato abstraktní metoda by měla vracet jako výsledek instanci podtřídy třídy `PluginResult`.

2.5 Shrnutí

Většina této kapitoly byla věnována popisu rodiny protokolů SSL/TLS a útokům na tyto protokoly. Když chce klient komunikovat se serverem zabezpečenou cestou, je třeba nejprve pomocí SSL/TLS HP protokolu sjednat atributy pro bezpečnou komunikaci. Mezi tyto atribudy patří:

- autentizace serveru (je možné i bez autentizace, ale není to doporučováno)
- autentizace klienta, pokud je to serverem vyžadováno
- sjednání společného tajemství
- sjednání klíčů pro šifrování
- sjednání šifrové sady
- sjednání kompresního algoritmu

Po dohodnutí a ověření těchto atributů je navázána bezpečná komunikace mezi klientem a serverem.

I přes takto nastavenou bezpečnou komunikaci, existují útoky, které dokáží zjistit tajemství obsažená v zašifrovaných zprávách. Výše byly popsány čtyři útoky z široké škály možných útoků, které lze využít k získání tajných informací (např.: cookie). Těmi útoky jsou CRIME, BEAST, POODLE a Lucky13. Tyto útoky mají společné to, že cílí na bezpečnou komunikaci zajištěnou protokoly SSL/TLS. Z této čtveřice se nejvíce odlišuje útok CRIME, který využívá kompresi dat pro zjištění tajné informace, je tedy jedno, jestli je pro

zabezpečenou komunikaci využit jako protokol SSL 3.0, nebo TLS 1.2. Zbylé tři utoky, BEAST, POODLE a Lucky13, využívají zranitelnosti v šifrovacím módu CBC. Každý z nich, ale cílí na jinou verzi protokol SSLT/TLS. BEAST využívá predikovatelnosti inicializačního vektoru u protokolu SSL 3.0 a TLS 1.0. POODLE využívá, že v současné době lze stále navázat komunikaci se serverem za pomoci protokolu SSL 3.0, který tvoří výplň zprávy nedeterministicky. Nakonec útok Lucky13 se od útoku POODLE moc neliší, ale narozdíl od útoku POODLE útočí na protokoly TLS 1.1, TLS 1.2 a DTLS. Nicméně je ze všech útoků časově nejnáročnější, jelikož vyžaduje mnoho navázání spojení se serverem, což není nejrychlejší.

Poslední útok popsaný v této kapitole je velice odlišný. Jedná se o útok Shellshock, který využívá chyby v „Bash“, což je součást většiny operačních systémů serverů. Ačkoliv se jedná o hodně rozdílný útok oproti útokům zmíněných dříve, byl tento útok studován, protože se jedná o aktuální a velmi častý útok.

Na konci této kapitoly jsou zmíněny programy, které testují servery na zranitelnosti vůči útokům, z nichž některé jsou zmíněny v této kapitole. Byly zde uvedeny webové stránky, které umožňují otestovat jakýkoliv server na internetu, ale i programy, které si může uživatel stáhnout a spouštět ze svého počítače. Nejvíce je však zaměřena na program SSLyze, který je dále v práci využíván při implementaci testovacího softwaru.

Kapitola 3

Návrh

Tato kapitola se věnuje návrhu testovacího programu pro příkazovou řádku platformy GNU/Linux. Tento program bude testovat server na zranitelnosti vůči útokům popsaných v předešlé kapitole. Po dohodě s vedoucím diplomové práce nebude vytvářen nový testovací program, ale bude využit některý ze stávajících programů na testování serverů. Pro vybraný program jsou v této sekci popsány zásuvné moduly, které rozšíří funkčnost vybraného programu.

Při konzultacích s vedoucím práce byl pro rozšíření funkčnosti zvolen program SSLyze. Tento program je blíže popsán v předešlé kapitole. Díky jeho struktuře je snadné jeho rozšíření o další funkcionality. Z popisu jeho struktury je zřejmé, že bude ideální pro každý útok vytvořit zásuvný modul, který bude testovat server, zda splňuje podmínky na provedení útoku. Zásuvné moduly se nesnaží provést útok jako takový, ale z informací získaných při navazování spojení se serverem určit, zda by mohl být daný server daným útokem napaden.

Největší problém je s útokem Shellshock. Jelikož SSLyze je program určený primárně na testování bezpečnosti SSL/TLS spojení, není zásuvný modul pro tento útok zcela vhodným rozšířením pro SSLyze. Pro tento útok je postačující podpora HTTP ze strany serveru. Nicméně tento útok je možný i při využití zabezpečeného HTTPS protokolu. Z těchto důvodů byly možné dvě varianty řešení, buď bude vytvořen zásuvný modul pro SSLyze, kde bude záležet na uživateli, zda tento test využije, nebo bude vytvořen samostatný program pouze na testování možnosti útoku Shellshock. Po dohodě s vedoucím byla nakonec zvolena druhá varianta, tedy pro Shellshock bude vytvořena samostatná aplikace, která bude testovat uživatelem zadaný server na tuto zranitelnost. Testovací nástroje pro zbylé útoky budou po dohodě s vedoucím navrženy jako rozšiřující moduly do programu SSLyze.

3.1 Možné knihovny k použití

Pro vytvoření SSL/TLS spojení je potřeba mít nainstalovanou SSL knihovnu. V operačních systémech Linux slouží pro instalaci SSL knihovny modul OpenSSL. Tato knihovna může být naistalována jako součást jiného programu. V takovém případě je však knihovna nastavena v základním nastavení od vyvojářů OpenSSL, nebo podle požadavků aplikace, která modul OpenSSL do operačního systému nainstalovala.

Modul OpenSSL a s ním spojenou knihovnu SSL lze nainstalovat do operačního systému manuálně. Při takové instalaci lze nastavit podporované verze protokolů SSL/TLS, povolit používání komprese při komunikaci, povolit šifrové sady, které při základní instalaci jsou zakázány. V závislosti na konfiguraci lze určovat bezpečnost modulu OpenSSL, který je instalován na daný operační systém. Pro návrh zásuvných modulů pro SSLyze je vhodné provést konfiguraci OpenSSL tak, aby klient podporoval všechny verze protokolů SSL/TLS a šifrové sady, které jsou pro SSL/TLS navazování spojení dostupné.

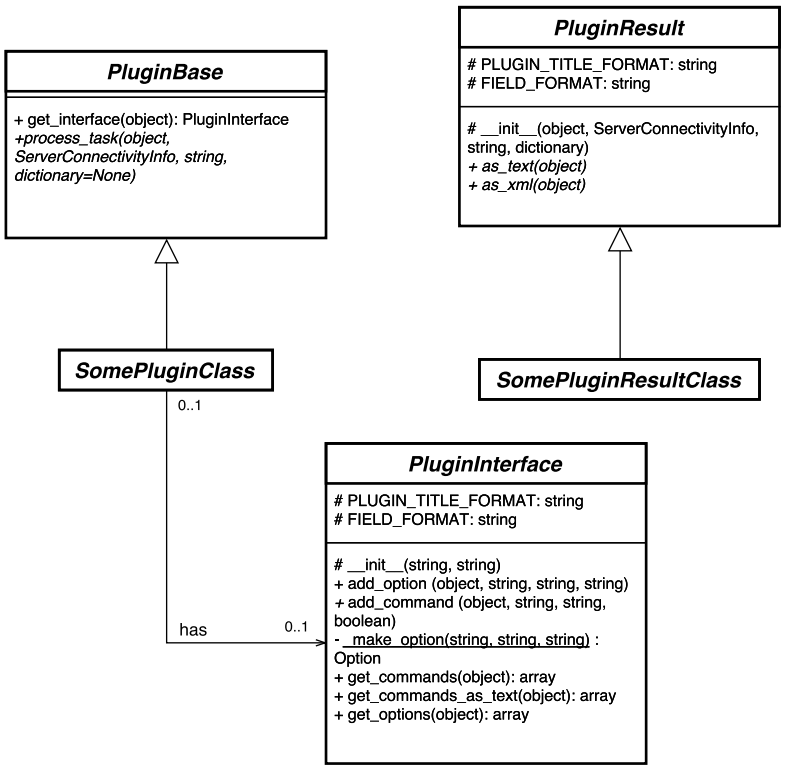
Pro práci se SSL knihovnou slouží v jazyce Python dodatečné knihovny. Tyto knihovny mají za úkol poskytnout funkce i parametry knihovny SSL pro tento programovací jazyk. V programu SSLyze je využívána knihovna nassl, která je speciálně implementována pro potřeby programu SSLyze. Další používanou knihovnou pro komunikaci s openssl je balíček pyopenssl. Díky této knihovně je umožněn snadný přístup ke knihovně SSL. Návrhy zásuvných modulů jsou vytvořeny tak, že je počítáno s využitím právě zmíněných knihoven pro práci se SSL.

3.2 Návrhy zásuvných modulů

Tato sekce popisuje návrh jednotlivých zásuvných modulů pro testování serveru na zranitelnosti vůči útokům BEAST, CRIME, POODLE a Lucky13. V jednotlivých podsekcích je popsán návrh a funkční požadavky jednotlivých zásuvných modulů a aplikace. Jsou zde popsány i obecné funkční požadavky na všechny zásuvné moduly.

Každý zásuvný modul bude obsahovat dvě třídy po vzoru již implementovaných zásuvných modulů. První třída bude mít za úkol zjistit zda je server zranitelný na daný útok. Tato třída bude dědit od abstraktní třídy `PluginBase`, a bude obsahovat instanci třídy `PluginInterface`, díky které SSLyze identifikuje zásuvný modul a získává o něm bližší informace. Součástí třídy je implementace abstraktní funkce `process_task`, která zajišťuje provedení otestování serveru a vyhodnocení zranitelnosti. Funkce vrací instanci podtřídy `PluginResult`. Každý zásuvný modul obsahuje implementaci podtřídy `PluginResult`, která slouží k poskytování výsledků testování serveru daným zásuvným modulem. Součástí každé třídy pro výpis výsledků zásuvného modulu jsou dvě funkce. První funkce, `as_text`, vytváří výstup pro výpis na obrazovku

příkazové řádky. Druhá funkce, `as_xml`, vytvaří výstup pro XML soubor. Vizte obr. 3.1, kde je zobrazena základní struktura, od které dědí jednotlivé třídy zásuvných modulů.



Obrázek 3.1: Struktura tříd pro zásuvné moduly pro SSLyze.

Dále bude vytvořen soubor s implementací tříd a konstant pro práci s šifrovými sadami, které půjde vložit do potřebných zásuvných modulů. Jedna třída bude reprezentovat přijatou šifrovou sadou serverem, zatímco druhá bude reprezentovat šifrovou sadu, kterou server zamítl. Konstanty obsažené v tomto souboru budou sloužit pro překlad názvů protokolů SSL/TLS na hodnoty, které je reprezentují dále při navazování spojení, a pro překlad zkrácených jmen šifrových sad na jejich úplný název.

3.2.1 Funkční požadavky

Navrhované zásuvné moduly budou splňovat následující funkční požadavky:

- Zjištění potřebných informací ze serveru
- Na základě zjištěných infodmací vyhodnotit míru zranitelnosti
- Umožnit výpis řešení na obrazovku příkazové řádky

- Umožnit výpis řešení do souboru XML

Výše zmíněné požadavky jsou obecnými požadavky na všechny zásuvné moduly bez ohledu na to, co je jejich úkolem. Jednotlivé zásuvné moduly budou mít své funkční požadavky specifikovány dále.

3.2.2 Zásuvný modul BEAST

Tento zásuvný modul bude sloužit pro otestování serveru na zranitelnosti, které umožňují provedení útoku BEAST. Tento útok byl popsán v předchozí kapitole a na základě zjištěných informací o tomto útoku, jsou navrženy následující funkční požadavky:

- Ověření zda uživatelem zadaný server podporuje protokol SSL verze 3.0.
- Ověření zda uživatelem zadaný server podporuje protokol TLS verze 1.0.
- Zjistit všechny šifrové sady podporované klientem pro protokoly podporované serverem.
- Provést TLS navázání spojení se serverem s jednotlivými, možnými šifrovými sadami.
- Vyhodnotit míru zranitelnosti serveru a uložit výsledek pro výpis.

Vstupem pro zásuvný modul jsou informace pro navázání spojení se serverem. Za využití těchto informací je provedeno ověření podpory jednotlivých protokolů. Z popisu útoku, tak jak je to popsáno v předchozí kapitole, je třeba zjistit zda server podporuje navázání spojení za využití protokolu SSL 3.0 nebo TLS 1.0. Pokud bude zjištěno, že jeden nebo oba protokoly jsou serverem podporovány, zjistí si zásuvný modul všechny šifrové sady, které jsou pro daný server k dispozici a provede pokusy o navázání spojení se server.

Zásuvný modul upraví každou ClientHello zprávu HP tak, aby seznam podporovaných šifrových sad obsahoval vždy jen jednu šifrovou sadu ze všech možných. Pokud server umožňuje komunikaci za využití šifrové sady, která byla poslána v ClientHello zprávě, potom dojde k bezproblémovému navázání spojení SSL/TLS. V tomto případě si zásuvný modul uloží danou šifru do seznamu serverem podporovaných šifrových sad. Pokud server šifrovou sadu, zaslanou v ClientHello zprávě, nepodporuje, zašle klientovi chybovou zprávu a ukončí spojení s klientem. Když dojde k tomuto případu, je šifrová sada přidána do seznamu serverem nepodporovaných šifrových sad.

Po rozdělení všech možných šifrových sad do skupin podporovaných a nepodporovaných serverem, přijde na řadu vyhodnocení míry zranitelnosti. Zásuvný modul projde seznam serverem podporovaných šifrových sad a hledá šifrové sady, které obsahují v názvu CBC. Tyto šifrové sady totiž využívají jako šifrovací operační mód CBC. Právě tyto šifrové sady jsou zranitelné na

útok BEAST. Pokud zásuvný modul nalezne v seznamu serverem podporovaných šifrových sad takovouto šifrovou sadu, je tato šifrová sada přidána do seznamu zranitelných šifrových sad. Pokud není seznam zranitelných šifrových sad prázdný, je daný server označen za zranitelný. Pokud server nepodporuje protokol SSL 3.0, ani TLS 1.0, nebo podporuje alespoň jeden z protokolů, ale nepodporuje šifrové sady s opračním módem CBC, pak je tento server označen jako bezpečný proti útoku BEAST.

3.2.3 Zásuvný modul CRIME

Tento zásuvný modul bude sloužit pro testování serveru na zranitelnosti, které využívá útok CRIME. Útok je popsán v předešlé kapitole a z tohoto popisu jsou odvozeny funkční požadavky na tento zásuvný modul, kterými jsou:

- Navázat spojení s daným serverem pomocí všech protokolů SSL/TLS
- Zkontrolovat zda navázané spojení používá kompresi
- Vyhodnotit míru zranitelnosti a uložit výsledek

Jelikož útok CRIME je nezávislý na verzi použitého protokolu pro bezpečné spojení, jsou testovány všechny verze. Nikde totiž není řečeno, že server, který nepodporuje kompresi pro protokol SSL 3.0, nepodporuje kompresi i pro ostatní verze protokolů SSL/TLS. To je důvod, proč je navazováno spojení s jednotlivými verzemi protokolů SSL/TLS od verze SSL 3.0 po TLS 1.2.

Server je označen za zranitelný, v případě zjištění použití komprese u navázaného spojení. Pokud je alespoň u jedné verze protokolu zjištěno použití kompresního algoritmu pro zabezpečenou komunikaci, je daný server označen jako zranitelný. Server není zranitelný, pokud ani u jednoho z možných protokolů, není pro zabezpečenou komunikaci použita komprese.

3.2.4 Zásuvný modul POODLE

Tento zásuvný modul bude sloužit pro testování serveru na zranitelnosti, které využívá útok POODLE. Na základě popisu tohoto útoku v předchozí kapitole, byly dány následující funkční požadavky:

- Ověřit zda server podporuje protokol SSL 3.0
- Získat všechny šifrové sady podporované protokolem SSL 3.0
- Ověřit podporu jednotlivých šifrových sad ze strany serveru
- Vyhodnotit míru zranitelnosti a uložit výsledek

Útok POODLE využívá zranitelnosti v implementaci tvorby výplně zprávy pro šifrování v protokolu SSL 3.0. Z tohoto důvodu zásuvný modul nejprve

ověří, zda uživatelem zadaný server podporuje navázání spojení s využitím protokolu SSL 3.0. Pokud je možné navázat spojení s využitím protokolu SSL 3.0, jsou dále ověřovány šifrové sady. V opačném případě je server považován za bezpečný vůči útoku POODLE.

V případě podpory protokolu SSL 3.0 jsou pro každou šifrovou sadu vytvořeny zprávy ClientHello. Pomocí této zprávy jsou navazována spojení se zadaným serverem a tím je ověřena podpora šifrové sady, která je poslána ve zprávě ClientHello. Pokud je šifrová sada podporována je navázáno spojení se zvolenou šifrovou sadou. V opačném případě server zašle klientovi zprávu s chybovou hláškou. Takto je postupně vytvořen seznam podporovaných a nepodporovaných šifrových sad. Následně je ověřeno, zda v seznamu podporovaných šifrových sad je nějaká, která používá pří šifrování operační mód CBC.

Nakonec dojde k vyhodnocení zranitelnosti daného serveru. Pokud server nepodporuje protokol SSL 3.0, nebo podporuje protokol 3.0, ale nepoužívá šifrové sady s operačním módem CBC, tak je daný server označen jako bezpečný proti útoku POODLE. V případě, že server podporuje navázání spojení s využitím protokolu SSL 3.0 a zároven používá šifrové sady s operačním módem CBC, je daný server označen jako zranitelný.

3.2.5 Zásuvný modul Lucky13

Tento zásuvný modul bude testovat server na zranitelnosti, které využívá útok Lucky13. Z popisu útoku v předešlé kapitole je odvozeno, že tento zásuvný modul bude mít dvě možnosti spuštění. Test serveru bude proveden buď pomocí protokolu TCP, tedy klasická kontrola protokolu SSL/TLS, nebo pomocí protokolu UDP, tedy bude ověřována bezpečnost protokolu DTLS. Vzhledem k tomuto faktu byly navrhnuty i následující funkční požadavky:

- Ověřit, zda server podporuje protokol TLS 1.1
- Ověřit, zda server podporuje protokol TLS 1.2
- Ověřit, zda server podporuje protokol DTLS 1.0
- Ověřit, zda server podporuje protokol DTLS 1.2, pokud je podporován klientem
- Získat všechny podporované šifrové sady pro daný protokol
- Zjistit, které šifrové sady podporuje server
- Vyhodnotit míru zranitelnosti a uložit výsledek

Zásuvný modul si nejprve ověří podporu protokolů TLS 1.1 a TLS 1.2. Útok Lucky13 je nadstavbou k útoku POODLE, jelikož ve vyšších verzích

TLS protokolu byly odstraněny chyby v implementaci, které POODLE využíval. Nicméně stále dochází k časovým únikům informace. Pokud tedy server nepodporuje protokoly TLS 1.1, ani TLS 1.2 dá se označit jako bezpečný proti útoku Lucky13, ale jen z důvodu, že v nižších verzích protokolů SSL/TLS lze využít jiných chyb v implementaci, které prozradí potřebnou informaci v kratším časovém období. Pokud server podporuje protokoly TLS 1.1, anebo TLS 1.2, jsou ověřovány šifrové sady.

Pro každou šifrovou sadu ze seznamu možných šifrových sad pro daný protokol je vytvořena zpráva ClientHello. Je provedeno navázání spojení se serverem a na základě odpovědí ze strany serveru je vytvořen seznam podporovaných a nepodporovaných šifrových sad. Pokud server s klientem naváže bezpečnou komunikaci, je daná šifrová sada přidána do seznamu podporovaných šifrových sad. Pokud server na poslanou zprávu ClientHello s danou šifrovou sadou odpoví chybovou zprávou, je tato šifrová sada přidána do seznamu nepodporovaných šifrových sad.

Po zjištění, které šifrové sady server podporuje, je tento seznam prověřen na zranitelné šifrové sady. Mezi zranitelné šifrové sady patří všechny, které pro šifrování používají operační mód CBC. Pokud je takováto šifrová sada nalezen v seznamu podporovaných šifrových sad, je server označen za zranitelný, jinak je považován za bezpečný proti útoku Lucky13. Toto otestování bude zařazeno do základního otestování serveru programem SSLyze, které je spuštěno příkazem `--regular`.

Avšak útok Lucky13 může být proveden i přes protokol UDP. Protokol UDP využívá pro vytvoření zabezpečené komunikace protokol DTLS, který je obdobný jako protokoly SSL/TLS akorád pracuje při nespolehlyvém spojení. Aby bylo možné úplné ověřování bezpečnosti vůči útoku Lucky13, bude moci uživatel spustit program SSLyze s příkazem, který spustí test na protokol DTLS. Při této možnosti spuštění programu SSLyze bude povinný parametr obsahující port, na které běží služba využívající DTLS. Ověření bezpečnosti je totožné s oveřováním bezpečnosti protokolů TLS popsané výše, avšak namísto protokolů TLS 1.1 a TLS 1.2, jsou ověřovány protokoly DTLS 1.0 a DTLS 1.2.

3.3 Návrh aplikace na testování zranitelnosti Shellshock

Jelikož tento útok je úplně jiný než ty předchozí, nehodí se jako rozšiřující modul pro SSLyze, který je zaměřen na testování bezpečnosti SSL/TLS spojení. Z tohoto důvodu byla po dohodě s vedoucím prace navržena samostatná aplikace, která bude testovat uživatelem zadaný server na zranitelnost vůči útoku Shellshock. Pro tuto aplikaci byly dány funkční požadavky vzhledem k popisu útoku v předhozí kapitole. Těmi funkčními požadavky jsou:

- Kontrola, zda lze navázat spojení na daný server

- Vytvoření nebezpečné hlavičky HTTP/HTTPS zprávy
- Zaslání HTTP/HTTPS dotazu na server
- Kontrola odpovědi ze strany serveru
- Vyhodnocení zranitelnosti a výpis výsledku

Útok Shellshock cílí na HTTP případně na HTTPS servery, které používají na svých stránkách CGI skripty. Jelikož HTTP i HTTPS servery mohou běžet na různých portech, je třeba identifikovat, kdy je testován HTTP server a kdy je testován HTTPS server. Toto určení bude ležet na uživateli aplikace, který bude muset sdělit pomocí přepínače, zda hodlá testovat HTTPS server, protože aplikace bude automaticky předpokládat, že je testován HTTP server.

K otestování, zda server je zranitelný vůči útoku Shellshock, neexistuje jiná možnost, než samotné provedení útoku. Toto je velice choulostivá situace, protože je třeba ověřit zda je server zranitelný, ale zároveň nesmí dojít k jeho poškození. Pro testováni bylo vybíráno mezi variantou vložení příkazu `ping`, který by zaslal jeden ICMP paket na IP adresu, odkud byl odeslán dotaz HTTP/HTTPS, a příkazu `echo`, který by provedl výpis zadaného textu do HTTP/HTTPS odpovědi, do hlavičky HTTP/HTTPS dotazu.

Nakonec byla zvolena druhá varianta, z důvodu snadnější implementace i vyhodnocení zranitelnosti. Pro první variantu by totiž bylo zapotřebí spustit další program na pozadí. Tento program by musel být spuštěn s maximálním oprávněním, aby mohl odposlouchávat komunikaci na určeném rozhraní. Odposlouchávací program by se musel sám ukončovat a předávat výsledky zpět do aplikace, kde by bylo provedeno vyhodnocení. Navíc pokud by byl testován server mimo lokální infrastrukturu příkaz `ping` by nemusel vždy fungovat. Z těchto důvodů byla zvolena varianta s příkazem `echo`, který vloží do HTTP/HTTPS odpovědi text zaslaný v hlavičce HTTP/HTTPS dotazu.

Aplikace bude ověřovat zranitelnost zasláním následující HTTP/HTTPS zprávy. Vytvoří si hlavičku HTTP nebo HTTPS dotazu, do které na pozici parametru `User-Agent` vloží následující sekvenci příkazů:

```
() { :; }; echo; echo "Your server is vulnerable to CVE-2014-6271
    (Shellshock). Please fix this as soon as possible"
```

Pokud aplikace obdrží od serveru jako odpověď zprávu, ve které se bude vyskytovat text „CVE-2014-6271“, potom je daný server zranitelný na útok Shellshock. Pokud odpověď od serveru tento text neobsahuje, pak server není zranitelný vůči útoku Shellshock.

3.4 Shrnutí

Kapitola se zabývá návrhem rozšiřujících modulů pro program SSLyze a zvláštního programu pro testování zranitelnosti serveru vůči útoku Shellshock. Zásuvné moduly jsou navrhovány po dohodě s vedoucím práce pro program

SSLyze namísto návrhu vlastního testovacího programu s výjimkou útoku Shellshock, pro který je navržen zvláštní testovací program. Záměrem návrhu zásuvných modulů pro testování zranitelností na útoky BEAST, CRIME, POODLE a Lucky13 je zvýšit použitelnost programu SSLyze, a zároveň usnadnění vyhodnocení zranitelnosti serveru na běžné síťové útoky.

Pro otestování SSL/TLS spojení je zapotřebí knihovny OpenSSL, která umožňuje vytvoření šifrovaného spojení. Dalšími knihovnami využitými při návrhu, jsou speciální knihovny pro jazyk Python, které umožňují přístup Python programu k funkcím a konstantám obsažených v OpenSSL knihovně. SSLyze využívá svou vlastní knihovnu nassl, která je vytvořena speciálně pro tento program. Další Python knihovna, která bude využívána pro navrhované zásuvné moduly bude pyOpenSSL.

Nakonec je zde popsána struktura navrhovaných zásuvných modulů a aplikace, a jejich funkční požadavky. Strukturu zásuvných modulů specifikuje program SSLyze. Ten obsahuje původní zásuvné moduly pro testování serverů, jejichž strukturu navrhované zásuvné moduly dodržují. Zásuvný modul obsahuje vždy dvě třídy. Jednu pro vykonávání testování serveru, která dědí od abstraktní třídy `PluginBase` a druhou pro reprezentaci výsledků, která je podtřídou třídy `PluginResult`. Do speciálního Python souboru jsou navrženy třídy pro přijaté šifrové sady serverem a pro zamítnuté šifrové sady serverem, a konstanty pro překlad názvu verze protokolu na hodnoty a zkráceného jména šifrové sady na úplný název.

Kapitola 4

Realizace

V této kapitole jsou popsány některé části implementace zásuvných modulů pro SSLyze a implementace aplikace pro testování zranitelnosti serveru na útok Shellshock. Je zde také popsáno jejich užití a testování.

4.1 Implementace zásuvných modulů pro SSLyze

Implementace zásuvných modulů byla provedena podle návrhů popsaných v předchozí kapitole. Během implementace bylo třeba vyřešit některé problémy pro správný běh programu SSLyze s navrhovanými zásuvnými moduly. Během implementace byla použita speciální knihovna pro program SSLyze, knihovna nassl, kterou využívají zásuvné moduly pro útoky BEAST, CRIME, POODLE a pro část zásuvného modulu pro útok Lucky13. Druhá část zásuvného modulu pro útok Lucky13, která testuje protokoly DTLS, je použita knihovna pyOpenSSL.

Knihovna pyOpenSSL je použita pro vytváření zabezpečeného UDP spojení. Pro vytvoření tohoto spojení se používa protokol DTLS, který má verze 1.0 a 1.2. Pro použití těchto protokolů bylo potřeba implementovat funkci, která nahrává z knihovny SSL metodu související s danou verzí protokolu DTLS. Jelikož se může stát, že klient nebude některou z verzí DTLS, nebo obě verze DTLS podporovat, jsou do pole `dtls_ary` uloženy pouze verze DTLS, které jsou dostupné v uživatelově knihovně SSL. Zdrojový kód této funkce je zobrazen na 4.1

Zdrojový kód 4.1: Funkce, kterou obsahuje `LuckyThirteen_vulnerability_tester_plugin.py` pro získání všech verzí DTLS protokolu podporované klientem.

```
def get_support_dtls_protocols_by_client(self):
    dtls_ary = []
    for dtls_version in self.DTLS_MODULE:
        try:
```

```
                SSL.Context._methods[dtls_version]=getattr(_lib, self.
                    DTLS_MODULE[dtls_version])
            except Exception as e:
                pass
            else:
                dtls_ary.append(dtls_version)
        return dtls_ary
```

Důležité bylo vyřešit propojení programu SSLyze a druhé části zásuvného modulu pro testování zranitelnosti serveru vůči útoku Lucky13 přes DTLS protokol. Jelikož program SSLyze je určen především pro testování serverů, které využívají protokoly SSL/TLS přes TCP protokol. Avšak útok Lucky13 je možné provést i přes zabezpečený protokol UDP pomocí protokolu DTLS. Program SSLyze před testováním zranitelností testuje zda jsou zadané inforormace o serveru platné. Provádí to tak, že se pokusí připojit na zadanou IP adresu, případně doménu, a port. Pokud se nepovede připojení, je tento server uložen do pole neplatných serverů zadaných uživatelem. V tomto bodě dochází k problému ve chvíli, kdy uživatel požaduje otestování serveru na zranitelnost vůči útoku Lucky13 přes protokol DTLS. Zadaný server nemusí využívat protokoly SSL/TLS s protokolem TCP. Aby bylo možné provést testování serveru, který nepoužívá protokoly SSL/TLS s protokolem TCP, přidal jsem do těla funkce `main` v souboru `sslyze_cli.py` řádky kódu (vizte ukázku 4.2), které před přidáním serveru mezi neplatné ověří, zda byl požadován test zranitelnosti na útok Lucky13 přes DTLS protokol. Pokud platí, že uživatel chtěl testovat zranitelnost serveru vůči útoku Lucky13 přes DTLS protokol, pak je tento server přidán do fronty pro otestování zásuvným modulem Lucky13. Takto je zajištěno, že zásuvný modul pro Lucky13, testující DTLS protokol, bude možné použít, i když server nepoužívá SSL/TLS. V případě, že uživatel chce například testovat server jak na zranitelnost vůči Lucky13 přes SSL/TLS, tak přes DTLS, a server nepoužívá SSL/TLS, potom je vrácena chybová hláška, která obsahuje informaci o tom, že se nepodařilo SSL/TLS spojení i s dovětkem, aby uživatel spustil program SSLyze pouze s příkazem `--lucky13_dtls`, pokud chce provést test zranitelnosti serveru vůči Lucky13 přes protokol DTLS.

Zdrojový kód 4.2: Část kódu v souboru `sslyze_cli.py` pro otestování serveru, který nepodporuje protokol SSL/TLS.

```
for tentative_server_info, exception in connectivity_tester.
    get_invalid_servers():
    if is_task_only_non_ssl_tests(args_command_list,
        available_commands):
        online_servers_list.append(tentative_server_info)
        for command in NON_SSL_COMMANDS:
            if getattr(args_command_list, command):
                plugin_options_dict = {}
                for option in available_commands[command].
                    get_interface().get_options():
```

```
                    if getattr(args_command_list, option.dest):
                        plugin_options_dict[option.dest]= getattr(
                            args_command_list, option.dest)
                plugins_process_pool.queue_plugin_task(
                    tentative_server_info, command,
                    plugin_options_dict)
    elif is_used_some_non_ssl_test_command(args_command_list):
        commands = get_use_non_ssl_commands(args_command_list)
        invalid_servers_list.append(('{}:{}'.format(
            tentative_server_info.hostname, tentative_server_info.
            port), ServerTestWithNonSSLCommandError(exception.
            error_msg,commands)))
    else:
        invalid_servers_list.append(('{}:{}'.format(
            tentative_server_info.hostname, tentative_server_info.
            port),
                                     exception))
```

Pro navržené zásuvné moduly bylo potřeba implementovat podrobný mód. Tento mód je voliteným příkazem pro SSLyze, který podrobně vypíše u navržených zásuvných modulů, co je na serveru testováno a s jakým výsledkem. Jelikož SSLyze vyžaduje i pro volitelné příkazy unikátnost, nelze u každého zásuvného modulu přidat příkaz pro podrobný mód. Nakonec jsem to vyřešil přidáním příkazu `--verbose` do společných příkazů pro SSLyze. Nyní však bylo ještě potřeba dostat tento příkaz do zásuvných modulů. Toto jsem zajistil přidáním následujícími řádky kódu do cyklu, který má za úkol vytvoření úloh pro zásvuný modul.

Zdrojový kód 4.3: Řádky kódu, které umožňuje běh zásuvného modulu v podrobném módu.

```
if args_command_list.verbose:
    plugin_options_dict['verbose'] = getattr(args_command_list,'
        verbose')
```

Část nově implementovaných zásuvných modulů jsem přidal do příkazu pro základní otestování uživatelem zadaného serveru. Přidal jsem do pole `REGULAR_CMD` v souboru `sslyze_cli.py` následující příkazy `beast`, `crime`, `poodle`, `lucky13_tls`. Díky tomu pro otestování serveru na útoky BEAST, CRIME, POODLE a Lucky13 stačí spustit program SSLyze s příkazem `--regular`. Jelikož program SSLyze je hlavně zaměřen na testování rodiny protokolů SSL/TLS, není součástí příkazu `--regular` příkaz pro provedení testu na zranitelnost vůči útoku Lucky13, při kterém je použit protokol DTLS.

4.2 Aplikace pro testování zranitelnosti Shellshock

Jelikož útok Shellshock je úplně odlišný od útoků BEAST, CRIME, POODLE a Lucky13, byl nástroj na testování zranitelnosti implementován jako samostatná aplikace. Aplikace je psána v jazyce Python, jelikož jsem chtěl dodržet,

aby všechny mnou implementované nástroje byly v jednom programovacím jazyce.

Aplikace je implementována tak, že obsahuje dva přepínače, které mohou být zadány uživatelem při spuštění aplikace. První přepínač `--https` určuje, zda uživatel chce testovat HTTPS server. Pokud je tento přepínač zadán, program navazuje zabezpečené spojení na HTTPS server, v opačném případě program předpokládá, že se připojuje na HTTP server. Druhým přepínačem je `--http_path`, který slouží k určení cesty k CGI skriptu na webovém serveru. Pokud není tento přepínač zadán, je automaticky použita cesta `/`.

Zadání cílového serveru může být provedeno více způsoby. Uživatel může zadat jméno domény nebo IP adresu serveru. Pokud chce ještě specifikovat, na jaký port má být provedeno HTTP nebo HTTPS spojení, potom za název domény nebo IP adresy napíše číslo portu, vizte ukázku použití v testování této aplikace. Uživatelem zadaná doména, popřípadě IP adresa, je od uživatelem zadaného čísla portu oddělena dvojtečkou. Pokud je zadána jen doména nebo IP adresa, je použito automaticky číslo portu 80 (HTTP). Pro získání cíle i přepínačů zadaných při spuštění aplikace je využit modul `optparse`. Aplikace neumožňuje testování více jak jednoho cíle najednou.

4.3 Testování

Funkčnost implementovaných zásuvných modulů pro program SSLyze a aplikace na testování zranitelnosti serveru vůči útoku Shellshok jsem ověřoval na serverech, které jsem si nainstaloval do virtuálního prostředí. Virtuální servery byly použity z důvodu, že nevlastním žádný fyzický server a provádět testování implementovaných zásuvných modulů a aplikace na jiných internetových serverech není vhodné.

První virtuální server běží s operačním systémem Ubuntu 14.04.4 LTS. Současně s instalací operačního systému byl naistalován i balík OpenSSL verze 1.0.1f z 6. ledna 2014. Poté jsem nastavil server tak, aby bylo možné navázat SSL/TLS spojení. Pro toto nastavení jsem vytvořil sebou podepsaný certifikát a povolil komunikaci serveru po portu 443 (HTTPS). Během testování jsem upravoval na serveru soubor `/etc/apache2/mods-available/ssl.conf`, který obsahuje nastavení SSL/TLS protokolů.

Jelikož tento server obsahuje novou verzi operačního systému Ubuntu, ve kterém jsou opravené zranitelnosti, které využívá útok Shellshock, vytvořil jsem ještě jeden virtuální server. Tento server běží se starou verzí operačního systému Ubuntu verze 6.06. Tato verze operačního systému obsahuje zranitelnosti, které využívá útok Shellshock, díky tomu je možné otestovat funkčnost aplikace pro testování zranitelnosti serveru na útok Shellshock.

4.3.1 Test zásuvného modulu BEAST

Abych otestoval správné vyhodnocování zranitelnosti zásuvným modulem BEAST, spustil jsem program SSLyze na různé konfigurace SSL/TLS na serveru. Nejprve jsem otestoval server na zranitelnost vůči útoku BEAST, v jeho základním nastavení. Server podporoval protokoly SSL 3.0 a vyšší a řetězec pro výběr šifrových sad byl `HIGH:MEDIUM:!aNULL:!MD5`. V tomto nastavení vyhodnotil program SSLyze, že server je zranitelný vůči útoku BEAST, vizte níže.

```
$ python sslyze_cli.py --beast 10.0.1.1

                                    ⋮

SCAN RESULTS FOR 10.0.1.1:443 - 10.0.1.1:443
---------------------------------------------

  * Vulnerability CVE-2011-3389:
                                          VULNERABLE - server is
                                              vulnerable to BEAST
                                              attack
      Support protocols: TLSv1, SSLv3
      Vulnerable cipher/ciphers:
        TLS_RSA_WITH_3DES_EDE_CBC_SHA
        TLS_RSA_WITH_SEED_CBC_SHA
        TLS_RSA_WITH_CAMELLIA_128_CBC_SHA
        TLS_DHE_RSA_WITH_AES_256_CBC_SHA
        TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
        TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA
        TLS_RSA_WITH_AES_256_CBC_SHA
        TLS_RSA_WITH_AES_128_CBC_SHA
        TLS_RSA_WITH_CAMELLIA_256_CBC_SHA
        TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
        TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
        TLS_DHE_RSA_WITH_SEED_CBC_SHA
        TLS_DHE_RSA_WITH_AES_128_CBC_SHA
        TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
        TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA


SCAN COMPLETED IN 1.03 S
------------------------
```

V druhém případě jsem nastavil server, tak aby nebyl zranitelný vůči útoku BEAST. Aby server nebyl zranitelný vůči útoku BEAST, nesmí server podporovat protokoly SSL 3.0 a TLS 1.0, nebo nesmí podporovat šifrové sady, které používají pro šifrování operační mód CBC. Rozhodl jsem se, že server nechám podporovat protokoly SSL 3.0 a TLS 1.0, ale zakáži v řetězci pro výběr šifrových sad všechny, které byly vypsány v předchozím testu. Výsledek testu takto nastaveného serveru můžete vidět níže.

```
$ python sslyze_cli.py --beast 10.0.1.1
```

```
                                    ⋮

SCAN RESULTS FOR 10.0.1.1:443 - 10.0.1.1:443
-------------------------------------------------------

  * Vulnerability CVE-2011-3389:
                                          OK - Not vulnerable to
                                              BEAST attack
      Support protocols: TLSv1, SSLv3
      Vulnerable cipher/ciphers:
        No vulnerable ciphers supported


SCAN COMPLETED IN 0.62 S
------------------------
```

4.3.2 Test zásuvného modulu CRIME

Tento zásuvný modul je také testován na virtuálním serveru. Server je automaticky nastaven tak, že nepodporuje SSL kompresi. Při tomto nastavení serveru musí SSLyze vypsat, že server není zranitelný, vizte níže.

```
$ python sslyze_cli.py --crime 10.0.1.1

                                    ⋮

SCAN RESULTS FOR 10.0.1.1:443 - 10.0.1.1:443
-------------------------------------------------------

  * Vulnerability CVE-2012-4929:
                                          OK - Not vulnerable to
                                              CRIME attack


SCAN COMPLETED IN 0.39 S
------------------------
```

Chtěl jsem otestovat i server, který by podporoval SSL kompresi. Bohužel se mi nepodařilo můj virtuální server nastavit tak, aby použil SSL kompresi, pokud ji klient umožňuje používat. Chyba není na klientově straně, protože odposlechem pomocí Wireshark ukázal, že klient ve zprávě ClientHello zasílá možnost použití DEFLATE komprese, ale server odpovídá ve zprávě ServerHello výběrem žádné komprese. Nicméně, podle odposlechu přes Wireshark a výpisu programu SSLyze při použití příkazu `--verbose`, jsem přesvědčen, že tento zásuvný modul dokáže odhalit server, který by byl zranitelný vůči útoku CRIME.

4.3.3 Test zásuvného modulu POODLE

Zásuvný modul pro otestování serveru na zranitelnost vůči útoku POODLE byl testován na virtuálním serveru. Virtuální server byl nastaven na podporu protokolu SSL 3.0 a podporované šifrové sady jsou nastaveny pomocí OpenSSL řetězce `HIGH:MEDIUM:!aNULL:!MD5`. Při tomto nastavení serveru je očekáván výsledek, že server je zranitelný na útok POODLE. Ukázka testu serveru je uvedena níže.

```
$ python sslyze_cli.py --poodle 10.0.1.1
                                    ⋮
SCAN RESULTS FOR 10.0.1.1:443 - 10.0.1.1:443
---------------------------------------------------

  * Vulnerability CVE-2014-3566:
                                        VULNERABLE - server is
                                            vulnerable to POODLE
                                            attack
    SSLv3 protocol:                       Is supported
      Vulnerable cipher/ciphers:
        TLS_RSA_WITH_3DES_EDE_CBC_SHA
        TLS_RSA_WITH_SEED_CBC_SHA
        TLS_RSA_WITH_CAMELLIA_128_CBC_SHA
        TLS_RSA_WITH_RC4_128_SHA
        TLS_DHE_RSA_WITH_AES_256_CBC_SHA
        TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
        TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA
        TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA
        TLS_RSA_WITH_CAMELLIA_256_CBC_SHA
        TLS_RSA_WITH_AES_128_CBC_SHA
        TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
        TLS_ECDHE_RSA_WITH_RC4_128_SHA
        TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
        TLS_DHE_RSA_WITH_SEED_CBC_SHA
        TLS_DHE_RSA_WITH_AES_128_CBC_SHA
        TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
        TLS_RSA_WITH_AES_256_CBC_SHA


SCAN COMPLETED IN 1.02 S
------------------------
```

Při druhém testu jsem nastavil server tak, aby byl bezpečný vůči útoku POODLE. Server bude bezpečný, jestliže nebude podporovat protokol SSL 3.0, který není považován za bezpečný, nebo server nepodporuje šifrové sady, které při šifrování používají operační mód CBC. Server jsem nastavil tak, aby nepodporoval protokol SSL 3.0. Výsledek testu provedený na takto nastavený server je uveden níže.

```
$ python sslyze_cli.py --poodle 10.0.1.1
```

```
                                    ⋮

SCAN RESULTS FOR 10.0.1.1:443 - 10.0.1.1:443
-------------------------------------------------------

 * Vulnerability CVE-2014-3566:
                                        OK - Not vulnerable to
                                            POODLE attack
   SSLv3 protocol:                      Is not supported


SCAN COMPLETED IN 0.09 S
-----------------------
```

4.3.4 Test zásuvného modulu Lucky13

Tento zásuvný modul byl vyzkoušen, jak pro protokoly TLS, tak pro protokoly DTLS. Nejprve jsem vyzkoušel správnou funkčnost zásuvného modulu při testování TLS protokolů. Zásuvný modul na testování zranitelnosti serveru vůči útoku Lucky13 prověřuje zda server používá protokol TLS 1.1, nebo TLS 1.2 a zároveň používá nějakou šifrovou sadu které využívá operační mód CBC. Virtuální server v základním nastavení podporuje obě zmíněné verze protokolu TLS a seznam podporovaných šifrových sad je nastaven OpenSSL řetězcem `HIGH:MEDIUM:!aNULL:!MD5`. Při tomto nastavení serveru je očekáván výstup, který bude upozorňovat na to, že je server zranitelný, vizte výpis níže.

```
$ python sslyze_cli.py --lucky13_tls 10.0.1.1

                                    ⋮

SCAN RESULTS FOR 10.0.1.1:443 - 10.0.1.1:443
-------------------------------------------------------

 * Vulnerability CVE-2013-0169:
                                        VULNERABLE - server is
                                            vulnerable to Lucky13
                                             attack
     Vulnerable cipher/ciphers:
       TLS_RSA_WITH_3DES_EDE_CBC_SHA
       TLS_RSA_WITH_SEED_CBC_SHA
       TLS_DHE_RSA_WITH_AES_256_CBC_SHA
       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
       TLS_RSA_WITH_CAMELLIA_256_CBC_SHA
       TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
       TLS_RSA_WITH_AES_128_CBC_SHA256
       TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA
       TLS_RSA_WITH_CAMELLIA_128_CBC_SHA
       TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
```

```
TLS_RSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
TLS_DHE_RSA_WITH_SEED_CBC_SHA
TLS_DHE_RSA_WITH_AES_128_CBC_SHA
TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA


SCAN COMPLETED IN 1.44 S
------------------------
```

Ve druhém testu zásuvného modulu je požadováno otestování serveru, který není zranitelný na útok Lucky13. Existují dvě varianty, kdy by měl zásuvný modul říci, že server není zranitelný na útok Lucky13. Pokud server nepodporuje použití protokolů TLS 1.1 a TLS 1.2, tak nelze provést útok Lucky13, který je časovým útokem právě na tyto protokoly. Druhý případ, kdy by měl být server označen za nezranitelný vůči útoku Lucky13, je, když server nepodporuje šifrové sady, které využívají operační mód CBC. Já jsem server nastavil podle druhého případu, protože se jedná o správnější nastavení serveru, tak aby byl co nejbezpečnější. Proto jsem zakázal šifrové sady, které byly vypsány jako zranitelné ve výše zmíněném výpisu. Níže je ukázka výpisu testu takto nastaveného serveru.

```
$ python sslyze_cli.py --lucky13_tls 10.0.1.1
                                    :

SCAN RESULTS FOR 10.0.1.1:443 - 10.0.1.1:443
---------------------------------------------

  * Vulnerability CVE-2013-0169:
                                        OK - Not vulnerable to
                                          Lucky13 attack

SCAN COMPLETED IN 0.47 S
------------------------
```

Nakonec jsem otestoval zda funguje zásuvný modul pro testování protokolů DTLS. Pro toto otestování jsem si na serveru vygeneroval nový certifikát a klíč. Po vygenerování těchto bezpečnostních parametrů jsem v rámci svého virtuálního server, spustil další server pomocí OpenSSL. OpenSSL server jsem spustil pomocí tohoto příkazu:

```
openssl s_server -dtls1 -cert cert.pem -key key.pem -port 4444
```

Poté jsem spustil program SSLyze, aby otestoval tento server. Jelikož je potřeba zadat port, na kterém server běží, má příkaz `--lucky13_dtls` povinný

další příkaz `--port`, ve kterém je tento port určen uživatelem. Výpis z testu je uveden níže.

```
$ python sslyze_cli.py --lucky13_dtls --port=4444 10.0.1.1

                                    ⋮

 SCAN RESULTS FOR 10.0.1.1:443 - 10.0.1.1:443
 ---------------------------------------------

SCAN IS DONE FOR DTLS PROTOCOLS - 10.0.1.1:4444
-----------------------------------------------

  * Vulnerability CVE-2013-0169:
                                          VULNERABLE - server is
                                              vulnerable to Lucky13
                                               attack
      Vulnerable cipher/ciphers:
        TLS_RSA_WITH_3DES_EDE_CBC_SHA
        TLS_RSA_WITH_SEED_CBC_SHA
        TLS_RSA_WITH_CAMELLIA_128_CBC_SHA
        TLS_DHE_RSA_WITH_AES_256_CBC_SHA
        TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
        TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA
        TLS_RSA_WITH_AES_256_CBC_SHA
        TLS_RSA_WITH_AES_128_CBC_SHA
        TLS_RSA_WITH_CAMELLIA_256_CBC_SHA
        TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
        TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
        TLS_DHE_RSA_WITH_SEED_CBC_SHA
        TLS_DHE_RSA_WITH_AES_128_CBC_SHA
        TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
        TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA


 SCAN COMPLETED IN 0.67 S
 ------------------------
```

4.3.5 Test aplikace pro testování zranitelnosti Shellshock

Testování aplikace, která byla implementována, aby testovala zranitelnost serveru vůči útoku Shellshock, je prováděno na dvou virtuálních serverech. První server používá starší zranitelný operační systém Ubuntu 6.06. Druhý server, který byl používán i v předchozích testech, používá novější opravený operační systém Ubuntu 14.04, který není zranitelný vůči útoku Shellshock. Na výpisu níže je nejprve testován server se starším, zranitelným operačním systémem, a poté server s novějším, bezpečným operačním systémem.

```
$ python ShellshockTester.py --http_path=/cgi-bin/example-cgi.sh
    10.0.2.1

  SCAN RESULT FOR 10.0.2.1:80
```

```
  ----------------------------------
    * Vulnerability CVE-2014-6271:
                                                    VULNERABLE - This
                                                        server is
                                                        vulnerable to
                                                        Shellshock attack

$ python ShellshockTester.py --https --http_path=/cgi-bin/example-
    cgi.sh 10.0.2.1:443

  SCAN RESULT FOR 10.0.1.1:443
  ----------------------------------
    * Vulnerability CVE-2014-6271:
                                                    OK - Not vulnerable
                                                        to Shellshock
                                                        attack
```

Kapitola 5

Závěr

Zadáním mé diplomové práce bylo vytvoření nástroje pro příkazovou řádku platformy UNIX/Linux pro testování uživatelem zadaného serveru na zranitelnosti vůči útokům BEAST, CRIME, POODLE, Lucky13 a Shellshock.

Nejdříve byla popsána rodina protokolů SSL/TLS. Popis byl zaměřen především na SSL 3.0, jelikož útoky, dále popsané v této práci, využívají zranitelností v tomto protokolu a také z důvodu, že protokoly TLS vychází z protokolu SSL 3.0. Byla zmíněna historie protokolu SSL a byla popsána jeho struktura, především jeho podprotokoly, ze kterých se protokol SSL skládá.

Dále byly popsány síťové útoky BEAST, CRIME, POODLE, Lucky13 a Shellshock. V rámci jednotlivých popisů útoků byly zmíněny základní informace o každém útoku, jako jsou údaje o tom, kdo daný útok prezentoval, nebo kdy byl útok prezentován. Také byl popsán základní princip jednotlivých útoků a zranitelnost, kterou daný útok využívá.

Na závěr analýzy byly popsány programy na testování serveru, které jsou na internetu dostupné. Podrobněji byl popsán program SSLyze, který byl používán v dalších částech mé diplomové práce. Byla popsána struktura programu SSLyze a jeho fungování s ohledem na jeho rozšíření o zásuvné moduly v dalších částech mé diplomové práce.

Po dohodě s vedoucím mé diplomové práce byly navrženy zásuvné moduly pro útoky BEAST, CRIME, POODLE a Lucky13, které rozšířily funkčnost programu SSLyze, a dále byla navržena samostatná aplikace na testování zranitelnosti vůči útoku Shellshock. Zásuvné moduly byly navrženy tak, aby vhodně a bezchybně rozšiřovaly funkčnost programu SSLyze.

Na základě návrhu byly implementovány jednotlivé zásuvné moduly a aplikace pro testování zranitelnosti serveru vůči útoku Shellshock. Implementace zásuvných modulů byla provedena v jazyce Python, jelikož je v tomto jazyce napsán i program SSLyze. Díky implementaci navržených zásuvných modulů bylo dosaženo rozšíření programu SSLyze. Takto rozšířený program SSLyze je snadněji použitelný pro odhalování zranitelností serveru na běžné síťové útoky. Testy na zranitelnost serveru vůči útokům BEAST, CRIME, POODLE

a Lucky13 byly přidány do základního otestování serveru, které je voláno pomocí přepínače `--regular`. U testu na zranitelnost serveru vůči útoku Lucky13 byla přidána jen část testu, které je vedena přes zabezpečené TCP spojení. Jako volitelná část testu byla přidána část zásuvného modulu pro testování zranitelnosti serveru vůči útoku Lucky13, který je veden na DTLS protokol. Jako samostatný testovací nástroj byla implementována aplikace pro testování zranitelnosti serveru vůči útoku Shellshock. Toto řešení bylo po dohodě s vedoucím uznáno za nejvhodnější s ohledem na to, že se jedná o výrazně rozdílný útok než jsou BEAST, CRIME, POODLE a Lucky13, a že by bylo z tohoto hlediska nevhodné rozšiřovat program SSLyze o test na takto rozdílný útok. Aby byli všechny mnou implementované nástroje napsány v jednom programovacím jazyku, je aplikace na testování zranitelnosti serveru vůči útoku Shellshock napsána také v jazyce Python.

Všechny mnou implementované zásuvné moduly a aplikace byly testovány. Implementované zásuvné moduly i aplikace správně odhalují zranitelné servery na dané útoky. Proto jsem byl schopen odhalit zranitelné servery na daný útok. Z tohoto důvodu plánuji kontaktovat výrobce programu SSLyze, zda umožní začlenění mnou implementované zásuvné moduly do nástroje.

Literatura

[1] Freier, A. O.; Kocher, P. C.; Karlton, P.: *The Secure Sockets Layer (SSL) Protocol Version 3.0 [online]*. Netscape Communications, 2011, [cit. 2016-02-23]. Dostupné z: `https://tools.ietf.org/html/rfc6101#section-1`

[2] McKinley, H. L.: SSL and TLS: A Beginners' Guide [online]. 2003, [cit. 2016-02-23]. Dostupné z: `https://www.sans.org/reading-room/whitepapers/protocols/ssl-tls-beginners-guide-1029`

[3] Castro-Castilla, A.: Traffic analysis of an SSL/TLS Session. 2014, [cit. 2016-04-12]. Dostupné z: `http://blog.fourthbit.com/2014/12/23/traffic-analysis-of-an-ssl-slash-tls-session`

[4] Dostálek, L.; kolektiv: *Velký průvodce protokoly TCP/IP: Bezpečnost*. Praha: Computer Press, druhé vydání, 2003.

[5] Dierks, T.; Freier, A. O.; Kocher, P. C.; aj.: *The TLS Protocol Version 1.0 [online]*. Netscape Communications, 1999, [cit. 2016-02-24]. Dostupné z: `https://tools.ietf.org/html/rfc2246`

[6] Dočkal, J.: Bezpečnost komunikace klient-server. *Data security management*, ročník 6, č. 1, 2002: s. 38–40.

[7] Oppliger, R.: *SSL and TLS*. Norwood: ARTECH HOUSE, 2009, ISBN 13 978-1-59693-447-4, s. 57–59.

[8] Rizzo, J.; Duong, T.: Here Come The $\oplus$ Ninja. 2011, [cit. 2016-04-12]. Dostupné z: `http://netifera.com/research/beast/beast_DRAFT_0621.pdf`

[9] Sarkar, P. G.; Fitzgerald, S.: Attacks on SSL a comprehensive study of BEAST, CRIME, TIME, BREACH, LUCKY 13 & RC4 biases. 2013, [cit. 2016-03-16]. Dostupné z: `https://www.nccgroup.trust/globalassets/our-research/us/whitepapers/ssl_attacks_survey.pdf`

[10] Bard, G. V.: Vulnerability of SSL to Chosen-Plaintext Attack. 2004, [cit. 2016-03-16].

[11] Rijmen, V.; Daemen, J.: *The Design of Rijndael AES - The Advanced Encryption Standard*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, ISBN 36-620-4722-5.

[12] Lórencz, R.: Blokové šifry, DES, 3DES, AES, operační módy šifer. [cit. 2016-03-16]. Dostupné z: https://edux.fit.cvut.cz/courses/BI-BEZ/_media/bez_n4.pdf

[13] TIM Trustworthy Internet Movement: SSL Pulse. Duben 2016, [cit. 2016-04-12]. Dostupné z: https://www.trustworthyinternet.org/ssl-pulse/

[14] Rizzo, J.; Duong, T.: The CRIME attack. [cit. 2016-04-12]. Dostupné z: https://docs.google.com/presentation/d/11eBmGiHbYcHR9gL5nDyZChu_-lCa2GizeuOfaLU2HOU/edit?pref=2&pli=1#slide=id.g1d134dff_1_222

[15] Acunetix: CRIME SSL/TLS attack. [cit. 2016-04-11]. Dostupné z: https://www.acunetix.com/vulnerabilities/web/crime-ssl-tls-attack

[16] Goodin, D.: Crack in Internet's foundation of trust allows HTTPS session hijacking. [cit. 2016-04-11]. Dostupné z: http://arstechnica.com/security/2012/09/crime-hijacks-https-sessions/

[17] Google Inc.: SPDY: An experimental protocol for a faster web. [cit. 2016-04-12]. Dostupné z: http://dev.chromium.org/spdy/spdy-whitepaper

[18] Kelsey, J.: Compression and Information Leakage of Plaintext. [cit. 2016-04-11]. Dostupné z: http://www.iacr.org/cryptodb/archive/2002/FSE/3091/3091.pdf

[19] Deutsch, P.: *DEFLATE Compressed Data Format Specification version 1.3 [online]*. 1996, [cit. 2016-04-11]. Dostupné z: https://tools.ietf.org/html/rfc1951

[20] Möller, B.; Duong, T.; Kotowicz, K.: This POODLE Bites: Exploiting The SSL 3.0 Fallback. 2014, [cit. 2016-04-12]. Dostupné z: https://www.openssl.org/~bodo/ssl-poodle.pdf

[21] Bright, P.: SSL broken, again, in POODLE attack. [cit. 2016-04-12]. Dostupné z: http://arstechnica.com/security/2014/10/ssl-broken-again-in-poodle-attack/

[22] Mikle, O.: POODLE útok na SSLv3. [cit. 2016-04-12]. Dostupné z: http://www.root.cz/clanky/poodle-utok-na-sslv3/

[23] Moeller, B.; Langley, A.: *TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks*. [cit. 2016-04-12]. Dostupné z: https://tools.ietf.org/html/draft-ietf-tls-downgrade-scsv-00

[24] Kokeš, J.; Lórencz, R.: Formátování a doplnění zpráv. [cit. 2016-04-12]. Dostupné z: https://edux.fit.cvut.cz/courses/MI-KRY/_media/lectures/09/prednaska9.pdf

[25] Jager, M.; Kratochvíl, P.: Hrozby pro dětské pokoje. *CHIP*, ročník 26, č. 3, 2016: s. 28–29, ISSN 1210-0684.

[26] AlFardan, N. J.; Paterson, K. G.: Lucky Thirteen: Breaking the TLS and DTLS Record Protocols. 2013, [cit. 2016-04-15]. Dostupné z: http://www.isg.rhul.ac.uk/tls/TLStiming.pdf

[27] Goodin, D.: "Lucky Thirteen" attack snarfs cookies protected by SSL encryption. [cit. 2016-04-15]. Dostupné z: http://arstechnica.com/security/2013/02/lucky-thirteen-attack-snarfs-cookies-protected-by-ssl-encryption/

[28] ImperialViolet: Lucky Thirteen attack on TLS CBC. [cit. 2016-04-15]. Dostupné z: https://www.imperialviolet.org/2013/02/04/luckythirteen.html

[29] Mary, C.: Shellshock Attack on Linux Systems - Bash. *International Research Journal of Engineering and Technology*, ročník 02, č. 08, 2015: s. 1322–1325, ISSN 2395-0056.

[30] Perlroth, N.: Security Experts Expect 'Shellshock' Software Bug in Bash to Be Significant. [cit. 2016-04-18]. Dostupné z: http://www.nytimes.com/2014/09/26/technology/security-experts-expect-shellshock-software-bug-to-be-significant.html?_r=0

[31] Graham-Cumming, J.: Inside Shellshock: How hackers are using it to exploit systems. [cit. 2016-04-18]. Dostupné z: https://blog.cloudflare.com/inside-shellshock/

[32] Vaughan-Nichols, S. J.: Shellshock: Better 'bash' patches now available. [cit. 2016-04-18]. Dostupné z: http://www.zdnet.com/article/shellshock-better-bash-patches-now-available/

[33] Fuchs, J.: Bash I. [cit. 2016-04-18]. Dostupné z: http://www.abclinuxu.cz/clanky/navody/bash-i

[34] Codenomicon: The Heartbleed Bug. [cit. 2016-04-18]. Dostupné z: http://heartbleed.com/

[35] Symantec: Arbitrary Code Execution. [cit. 2016-04-18]. Dostupné z: http://www.pctools.com/security-news/arbitrary-code-execution/

Příloha A

Seznam použitých zkratek

ACE Arbitary Code Execution

AES Advanced Encryption Standard

AP Alert Protocol

BEAST Browser Exploit Against SSL/TLS

CA Certificate Authority

CBC Cipher Block Chaining

CCSP Change Cipher Spec Protocol

CGI Common Gateway Interface

CRIME Compression Ratio Info-leak Made Easy

DOS Denial of Service

DTLS Datagram Transport Layer Protocol

HMAC Keyed-hash Message Authentication Code

HP Handshake Protocol

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

ICMP Internet Control Message Protocol

ID Identifikace

IV Initialization Vector

KEA Key Exchange Algorithm

MAC Message Authentication Code

POODLE Padding Oracle On Downgraded Legacy Encryption

RLP Record Layer Protocol

RSA Rivest Shamir Adleman

SSL Secure Sockets Layer

TCP Transport Layer Protocol

TCP/IP Transmission Control Protocol/Internet Protocol

TLS Transport Layer Security

UDP User Datagram Protocol

URL Uniform Resource Locator

Příloha B

Obsah přiloženého CD

- `Pavel Soukup Diplomova prace 2016`
  - `README.txt` — stručný popis obsahu CD
  - `MANUAL.txt` — stručný návod na spuštění implementovaných zásuvných modulů a aplikace
  - `SRC` — složka se zdrojovými kódy
    - `impl` — zdrojové kódy implementace
      - `Shellshock-Tester` — složka obsahuje *.py soubor pro spuštění aplikace na testování zranitelnosti serveru vůči útoku Shellshock
      - `SSLyze-plugins` — složka obsahuje *.py soubory, ve kterých je implementace mých zásuvných modulů pro program SSLyze
    - `thesis` — zdrojová forma práce ve formátu LaTeX
  - `DOC` — dokumentace zdrojového kódu
    - `Shellshock-Tester-doc` — dokumentace zdrojového kódu aplikace pro testování zranitelnosti serveru vůči útoku Shellshock
    - `SSlyze-plugins-doc` — dokumentace zdrojových kódů zásuvných modulů pro program SSLyze
  - `SSLyze` — adresář s programem SSLyze rozšířeným o nové zásuvné moduly

- `text` — text diplomové práce ve formátu PDF
  - `DP_Soukup_Pavel_2016.pdf`