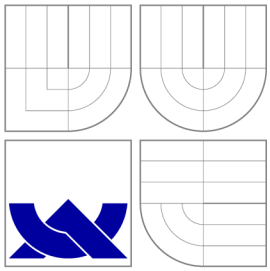
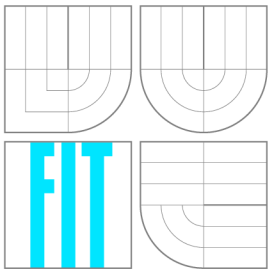


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AUTOMATIZACE MITM ÚTOKU PRO DEŠIFROVÁNÍ SSL/TLS

AUTOMATIZATION OF MITM ATTACK FOR SSL/TLS DECRYPTION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MAREK MARUŠIC
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN PLUSKAL
SUPERVISOR

BRNO 2016

Zadání bakalářské práce/18593/2015/xmarus05

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Marušic Marek**
Obor: Informační technologie
Téma: **Automatizace MitM útoku pro dešifrování SSL/TLS**
Automatization of MitM Attack for SSL/TLS Decryption
Kategorie: Počítačové sítě

Pokyny:

1. Seznamte se s SSL/TLS protokoly, jejich zranitelnostmi a známými útoky.
2. Seznamte se variací Man-in-the-middle útoků použitelných pro získání privátního klíče SSL/TLS sezení.
3. Navrhněte HW a SW sestavu s důrazem na mobilitu a možnost zachycení dat na disk. Využijte existující nástroje z bodu 2 a vyberte nejvhodnější, či sadu nejvhodnějších nástrojů, které zajistí, že uložený síťový provoz bude možné dále zpracovávat v nešifrované podobě.
4. Zhodnoťte kroky nutné k dosažení transparentního MitM. Zvažte, zdali by bylo možné cíleně podle znalosti použitého operačního systému dané kroky zjednodušit.
5. Otestujte řešení při experimentálním zapojení v laboratoři. Zaměřte se na zatížení zařízení, zda-li je schopné pracovat na rychlosti linky.

Literatura:

- Callegati, F., Cerroni, W. & Ramilli, M., Man-in-the-middle attack to the HTTPS protocol. IEEE Security and Privacy, 7(1), p.78-81. 2009.
- Dierks, T. & Rescorla, E., 2008. RFC 5246 - The transport layer security (TLS) protocol - Version 1.2. In *Network Working Group, IETF*. pp. 1-105.
- Wagner, D. & Schneier, B., 1996. Analysis of the SSL 3.0 protocol. In *Proceedings of the 2nd conference on Proceedings of the Second USENIX Workshop on Electronic Commerce - Volume 2*. USENIX Association, p. 4.
- Das, M.L. & Samdaria, N., 2014. On the security of SSL/TLS-enabled applications. *Applied Computing and Informatics*.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1, 2 a 3.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Pluskal Jan, Ing.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Protokoly SSL/TLS sú používané pre šifráciu sieťovéhej prevádzky. Poskytujú bezpečnú komunikáciu medzi klientmi a servermi. Komunikácia môže byť odpočúvaná pomocou MitM útoku. Táto práca je zameraná na atuomatizovanie MitM útoku a demonštráciu jej výsledkov. Automatizáciou sa zjednoduší spustenie útoku bez nutnosti študovania rôznych manuálových stránok a aby sa používatelia vyhli pracnej konfigurácii MitM zariadenia a mohli jednoducho zachytiť a analyzovať SSL/TLS komunikáciu. Automatizácia je vykonaná pomocou MitM sondy a python skriptu, ktorý nakonfiguruje sondu a spustí útok automaticky. Skript má jednoduché ovládanie, bez nutnosti špeciálnych znalostí. Skript zabezpečí konfiguráciu sondy, spustí nástroje pre záchyt sieťovej prevádzky do pcap formátu a na koniec spustí MitM nástroje potrebné pre MitM útok. Počas útoku sú obete varované klientskými aplikáciami o nebezpečnom pripojení. V tejto práci sa čitatelia môžu navyše dozvedieť o SSL/TLS protokoloch a možnosti ako odpočúvať dáta šifrované pomocou týchto protokolov.

Abstract

SSL/TLS are protocols used to encrypt network traffic. They provide secure communication between clients and servers. The communication can be intercepted with MitM attack. This paper is aimed to describe the automatization of MitM attack and demonstrate its results. The automatization is done by MitM probe and a python script, which configures the probe and starts the attack. The script is easy to use, without great effort. It takes care of configuration of the probe, then it starts the tools used for network traffic capture and at last it starts MitM tools to perform the attack. During the MitM attack, users are warned by client applications about insecure connection. The client applications either provide an option to establish a connection anyway or it forbids clients to establish the connection with insecure parameters. In this paper, the users can learn what are SSL/TLS protocols and about a possibility how to intercept the network traffic encrypted by these protocols.

Kľúčové slová

SSL, TLS, dešifrácia, MitM, Man-in-the-Middle, útok

Keywords

SSL, TLS, decryption, MitM, Man-in-the-Middle, attack

Citácia

MARUŠIC, Marek. *Automatizace MitM útoku pro dešifrování SSL/TLS*. Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Pluskal Jan.

Automatizace MitM útoku pro dešifrování SSL/TLS

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Jana Pluskala s použitím literatúry, ktorú uvádzam v zozname.

.....
Marek Marušic
19. mája 2016

Poďakovanie

Týmto by som sa chcel poďakovať svojmu vedúcemu Ing. Janovi Pluskalovi za odbornú pomoc, cenné rady a motiváciu počas vypracovávania bakalárskej práce.

Obsah

Kapitola 1

Úvod

V dnešnej dobe používa internet obrovské percento populácie. Pre niektorých ľudí je to neodmysliteľná súčasť života. Pomocou internetu sa ľudia dokážu spojiť, zistiť rôzne informácie, komunikovať spolu, zdieľať svoje zážitky a dáta, pomocou pár klikov na svojom telefóne, tablete či počítači. Pre zabezpečenie komunikácie medzi klientmi a aplikáciami sa používajú rôzne šifrovacie protokoly a algoritmy, čím sa obsah ich komunikácie zmení na nečitateľnú správu, ktorá pre tretie strany nedáva žiaden zmysel bez potrebných dešiforvacích parametrov.

Internet je samozrejme neoddeliteľnou súčasťou aj v komerčnej sfére, kde sa na pripojenie spoliehajú korporácie a pri jeho výpadku strácajú nemalé peniaze. Bežný ľudia si ani neuvedomujú koľko ich súkromných informácií je verejne dostupných a koľko ďalších informácií sa dá o nich získať, len kvôli ich ľahostajnosti a neznalosti, pomocou niektorých nástrojov. V niektorých prípadoch je dôležité dolovanie informácií zo sieťovej komunikácie za účelom chytenia zločincov a získanie dôkazov o rôznych kriminálnych činnostiach.

Pre šifráciu sieťovej komunikácie často používa SSL/TLS protokol, pre bezpečný prenos dát cez nezabezpečenú infraštruktúru Internetu. Avšak polícia alebo vláda neraz potrebuje dešifrovať a analyzovať komunikáciu medzi kriminálnikmi a podozrivými ľudmi. Existujúce nástroje, ktoré dokážu získať dešifrované dáta z SSL/TLS komunikácie sú náročné na použitie, preto by bežnému používateľovi trvalo veľmi dlho takýto útok vykonať. Navyše by musel používateľ použiť veľa rôznych nástrojov aby dokázal odpočúvať komunikácie v sieťach s rozličnými vlastnosťami (napr. šifrovacie sady, typy zapojenia do siete, použitie IPv4 alebo IPv6 atď.). Táto práca ma za úlohu poskytnúť jednoduché riešenie pre odpočúvanie SSL/TLS komunikácie, ktoré podporuje viacero spomínaných vlastností a zároveň je jednoduché pre použitie.

Kapitola 2 opisuje ako fungujú kryptografické protokoly SSL/TLS. Ďalej tu sú popísané niektoré protokoly, ktoré majú podobnú úlohu zašifrovať komunikáciu medzi klientom a serverom. V neposlednom rade je spomenutá možnosť dešifrovať SSL/TLS komunikáciu pomocou rôznych nástrojov a najvýznamnejšie chyby, ktoré vedú k dešifrácii SSL/TLS komunikácie.

Tretia kapitola nás prevedie analýzou dostupných open-source nástrojov. Tieto nástroje sa dajú použiť pri automatizácií scenárov vedúcich k dešifrácii SSL/TLS protokolov. Konkrétne tieto nástroje používajú MitM scenáre. Nástroje podporujú a ponúkajú rôzne možnosti, ktoré budú v tejto kapitole popísané. Ďalej budú nástroje porovnané, na základe ich schopností a je vybraný nástroj, ktorý bol použitý pri implementácii automatizácie.

V štvrtej kapitole je navrhnutá automatická konfigurácia sondy a nástrojov pre automatizáciu Man in the Middle scenáru. Sú tu popísane Linuxové aplikácie, ktoré je nutné

použiť aby bola konfigurácia zariadenia správna. Spomenutý je tiež návrh GUI a skriptu pre automatizáciu.

Piata kapitola sa venuje samotnej implementácii automatizácie. Popisuje konfiguráciu zvoleného operačného systému a nástrojov, ktoré je nutné použiť. V neposlednom rade tu je popísaný MitM Automatizér (skript pre automatizáciu MitM) a triedy, ktoré používa.

V šiestej kapitole bude možné si prečítať výsledky záťažových testov sondy, či dokáže pracovať na rýchlosti vyťaženej linky a aké straty paketov sa objavia. Na základe tohoto testovania sondy budeme môcť usúdiť, či je možné minimalizovať hárdverovú konfiguráciu čím by sa zmenšili rozmery sondy a bola jednoduchšie prenosná a lepšie maskovateľná. Bude tu tiež demonštrovaná samotná dešifrácia zachytených paketov pomocou získaných potrebných kľúčov v laboratórnych podmienkach.

Kapitola 2

Analýza protokolov SSL/TLS

Secure Sockets Layer (SSL) a *Transport Layer Security* (TLS) sú kryptografické protokoly používané pre šifráciu sieťovej prevádzky. Šifrovaním poskytujú bezpečnú komunikáciu medzi dvoma zariadeniami v sieti. SSL je predchodca TLS. S rapídnym rozširovaní *Internet*-u a *World Wide Web*-u bolo nutné zabezpečiť komunikáciu, ktorá obsahovala citlivé informácie určené práve len pre komunikujúce strany. [17]

V roku 1995 bol preto vytvorený internetový návrh s názvom *The SSL Protocol*. [9] Tento internetový návrh popisoval SSL protokol. Spoločnosť Netscape vyvinula SSL 1.0, táto verzia nebola nikdy zverejnená, kvôli veľkému množstvu bezpečnostných vád. SSL 2.0 bolo vydané v roku 1995, táto verzia tiež obsahovala množstvo bezpečnostných chýb čo viedlo k vydaniu verzie 3.0 v roku 1996. V dnešnej dobe sa kvôli bezpečnostným zraniteľnostiam tieto protokoly nepoužívajú. V roku 1999 bolo zverejnené RFC2246 [1], ktoré definuje protokol TLS 1.0. Neskôr v roku 2006 bolo definované TLS 1.1 a v roku 2008 bolo definované TLS 1.2 v roku 2016 bol vytvorený návrh TLS 1.3. Každá nová verzia prináša záplaty pre nájdené chyby a útoky počas používania jej predchodcu, čím prináša väčšiu bezpečnosť a odolnosť proti možným útokom.

V tejto kapitole budú podrobnejšie popísane protokoly SSL/TLS a taktiež niektoré používané šifrovacie protokoly pre bezpečnú komunikáciu po sieti. Okrem toho bude popísaný ich princíp niektoré pod-protokoly, ktoré sú v nich použité a naimplementované. V neposlednej rade tu budú popísané niektoré známe útoky na spomínané protokoly a zneužiteľné chyby, pomocou ktorých je možné dešifrovať prenášané dáta.

2.1 Protokol SSL/TLS

SSL a TLS sú kryptografické protokoly určené pre zabezpečenie komunikácie v nezabezpečenej infraštruktúre Internetu. V *Open Systems Interconnection* (OSI) modely [18] sa nachádzajú v 6. - prezentačnej vrstve, ktorá sa nachádza medzi transportnou a aplikačnou vrstvou. Toto je veľkou výhodou, pretože kvôli implementácií SSL/TLS nieje nutné meniť fyzickú infraštruktúru pripojenia, no taktiež nieje nutné upravovať 7. aplikačnú vrstvu čím napr. webové aplikácie nieje nutné upravovať kvôli použitiu SSL/TLS.

Sú to klient/server protokoly, ktoré poskytujú nasledovné bezpečnostné služby komunikujúcim stranám [14]:

- Overenie pripojených strán a pôvodu dát

[1] https://tools.ietf.org/html/rfc2246

- Dôvernosť pripojenia
- Integritu pripojenia

Bezpečnosť nieje jediný cieľ SSL a TLS, ale až 4 nasledovné hlavné ciele [16]:

Kryptografická bezpečnosť
Toto je hlavný problém a cieľ týchto protokolov. Ich úlohou je umožniť bezpečnú komunikáciu medzi dvoma stranami, ktoré si potrebujú vymieňať citlivé informácie, bez toho aby tretia strana bola schopná získať prístup k vymieňaným informáciám. Bezpečná komunikácia je poskytovaná pomocou šifrácie vymieňaných informácií.

Interoperabilita
Ďalším cieľom je zaistiť aby nezávislí programátori boli schopní vyvíjať programy, knižnice a aplikácie, ktoré sú schopné komunikovať medzi sebou pomocou bežných kryptografických parametrov. Jedná sa o schopnosť spolupracovať, poskytovať si služby a dosiahnutie vzájomnej súčinnosti rôznych systémov.

Rozšíriteľnosť
Dôležitý cieľ je nezávislosť na aktuálne použitých kryptografických metódach. Tento cieľ umožňuje jednoduchú migráciu medzi rôznymi šifrovacími algoritmami. Toto je veľmi dôležité aby pri migrácii nebolo nutné vytvárať nové protokoly.

Efektivita
Posledný cieľ je dosiahnuť predošlé ciele za akceptovateľného zníženia výkonu. Vysokú efektivitu je možné dosiahnuť pomocou redukcie náročných kryptografických operácií na minimum. Pomocou cache relácií sa dá tento počet operácií veľmi dobre znížiť.

SSL a TLS sa skladajú z nasledujúcich protokolov:

Record protokol
Používa sa pre bezpečný prenos dát protokolov vyššej vrstvy popísaných nižšie.

Handshake protokol
Je základný protokol SSL a TLS. Je zodpovedný za vyjednanie bezpečnostných atribútov relácie.

Change Cipher Spec Protocol
Umožňuje komunikujúcim stranám signalizovať zmeny v šifrovacích stratégiách. Protokol reprezentuje jediná správa, ktorá je zašifrovaná a komprimovaná pomocou aktuálne používanej šifrovacej stratégie. Správa je odoslaná oboma komunikujúcimi stranami pre oznámenie druhej strane, že nasledovné dáta budú šifrované pomocou novo vyjednaných šifrovacích pravidiel a kľúčov.

Alert Protocol
Slúži pre signalizáciu potenciálnych problémov a pre výmenu výstražných správ.

Application Data protokol
Je použitý pre bezpečný prenos aplikačných dát.[5]

2.1.1 SSL/TLS relácia

Pri každom pripojení pomocou SSL/TLS protokolu je vytvorené SSL/TLS relácia, ktorá uchováva potrebné informácie o danom pripojení. SSL/TLS relácia je zložená z nasledujúcich údajov:

identifikátor relácie Ľubovoľná sekvencia bajtov, vybraná serverom pre identifikáciu aktívneho alebo znovu použiteľného stavu relácie.

certifikát strany X509v3 certifikát komunikujúcej strany. Tento element môže byť prázdny.

kompresná metóda Algoritmus použitý pre kompresiu dát pred šifrovaním.

šifra Špecifikuje pseudonáhodnú funkciu pre generovanie kľúčov, šifrovací algoritmus dát a MAC algoritmus.

cipher spec Špecifikuje pseudonáhodnú funkciu používanú na vygenerovanie šifrovacích materiálov, algoritmus pre šifrovanie (ako napríklad AES atď) a MAC algoritmus. Definuje kryptografické atribúty napr. mac_length.

hlavné heslo 48-bajtové heslo zdieľané medzi klientom a serverom.

znovu použiteľnosť Indikuje možnosť použitia relácie pre nadviazanie nových spojení.

Z týchto údajov sa potom vytvoria bezpečnostné parametre použité record vrstvou pri ochrane aplikačných dát. Veľa spojení môže byť vytvorených pomocou rovnakej relácie pomocou funkcie obnovenia Handshake protokolu. [5]

2.1.2 Handshake protokol

Kryptografické parametre relácie sú dohodnuté počas TLS Handshake. Pri začatí prvej komunikácie serveru a klienta sa dohodnú na použitej verzii protokolu, použitých kryptografických algoritmoch, použitých zdieľaných kľúčoch a taktiež môžu voliteľne overiť identitu jeden druhého. Počas tejto komunikácie je použité asymetrické šifrovanie pomocou verejných kľúčov. Po úspešnom TLS handshake sú vygenerované zdieľané kľúče pomocou ktorých bude ďalšia komunikácia serveru a klienta šifrovaná symetrickým šifrovaním. V nasledujúcej tabuľke 2.1 je znázornený priebeh celého TLS Handshake.

2.1.3 Infraštruktúra verejných kľúčov

Infraštruktúra verejných kľúčov, často označovaná skratkou PKI (z anglického názvu Public Key Infrastructure), je systém digitálnych certifikátov, certifikačných úradov a ďalších registračných úradov, ktoré slúžia k verifikácií a overeniu platnosti všetkých strán zúčastnených v určitej elektronickej transakcii, pričom je použitá asymetrická kryptografia.[3]

Certifikát je digitálny dokument, ktorý uchováva verejný kľúč, jeho dobu platnosti, informácie o vlastníkovi certifikátu a digitálny podpis vydavateľa resp. overovateľa certifikátu. Tento dokument slúži na preukázanie vlastníctva verejného kľúča. Pomocou verejného kľúča sa overuje pri pripájaní identita serveru/clienta a umožňuje šifrovanú komunikáciu. Vydavateľ a podpisovateľ certifikátu sa v Public Key Infrastrucure (PKI) nazýva Certifikačná autorita (CA). CA je dôveryhodná inštitúcia, ktorá spravuje certifikáty (podpisovanie, revokovanie, vydávanie certifikátov, ...) a tiež vlastní digitálny certifikát. Pokiaľ dôverujeme tejto CA, môžeme vložiť jej certifikát do úložiska dôveryhodných certifikátov zariadenia.

Client		Server
ClientHello	$\longrightarrow$	
	$\longleftarrow$	Server hello
	$\longleftarrow$	Server certificate*
	$\longleftarrow$	Server key exchange*
	$\longleftarrow$	Certificate request*
	$\longleftarrow$	Server hello done
Client certificate*	$\longrightarrow$	
Client key exchange	$\longrightarrow$	
Certificate verify*	$\longrightarrow$	
[Change cipher spec]	$\longrightarrow$	
Finished	$\longrightarrow$	
	$\longleftarrow$	[Change cipher spec]
	$\longleftarrow$	Finished
Application Data	$\longleftrightarrow$	Application Data

Tabuľka 2.1: Handshake priebeh [5]
* Znázorňuje nepovinné správy alebo správy odosielané v závislosti na danej situácii.
[Change cipher spec] Nieje správa posielaná Handshake protokolom ale samostatný protokol spomínaný vyššie.

Certifikáty niektorých CA sú naimportované už v samotnom OS zariadenia a v jadre niektorých aplikácií (napr. vo webových prehliadačoch), ktoré dôverujú daným CA.

Je možné si vytvoriť vlastný certifikát pomocou rôznych nástrojov napr. OpenSSL[2] no po vytvorení je potrebné overiť vlastníka certifikátu a podpísať certifikát, čo môže uskutočniť CA (väčšinou oficiálne CA účtujú nejaký poplatok). Pomocou OpenSSL je taktiež možné si vytvoriť vlastnú CA, no vo webových prehliadačoch a aplikáciach, ktoré chcú dôverovať tejto CA je nutné naimportovať jej certifikát, čím potvrdíme dôveryhodnosť všetkých certifikátov podpísaných touto CA.

Ku každému verejnému kľúču je vygenerovaný privátny kľúč, ktorý je použitý pri asymetrickej kryptografii a pre podpisovanie vydávaných certifikátov. Tento kľúč musí byť veľmi dobre zabezpečený aby sa poň nikto okrem vlastníka nedostal. Pri sprenevere alebo odcudzení tohoto certifikátu je nutné tento certifikát ihneď revokovať a vytvoriť nový. Ak by sa ďalej používal odcudzený certifikát, celá certifikačná reťaz by bola kompromitovaná. Všetka komunikácia šifrovaná certifikátmi podpísanými podriadenými autoritami by sa dala dešifrovať pomocou odcudzeného certifikátu.

Koreňová CA vlastní certifikát, ktorý je podpísaný sebou samým [7]. Tento certifikát je použitý na digitálne podpisovanie ďalších certifikátov podriadených autorít. Podriadené autority môžu podpisovať certifikáty ďalších podriadených autorít alebo môžu podpisovať certifikáty jednotlivých hostiteľov. Takýmto spôsobom je vytvorená reťaz resp. strom certifikátov a autorít.

2.1.4 HTTP pomocou SSL/TLS

HTTP [6] je protokol určený pre distribuované, kolaboratívne a hypermediálne systémy. Používa ho World-Wide-Web po celom svete. Nepoužíva žiadne kryptografické ani šifrovacie prvky, čo umožňuje všetkým vidieť a upravovať obsah prenášaných dát. So zvyšujúcim

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

sa počtom aplikácií, ktoré prenášajú citlivé dáta pomocou HTTP a zvyšujúcim sa počtom pripojených jedincov bolo nutné zabezpečiť prenášané dáta. SSL/TLS je určené práve k tomuto zabezpečeniu. Každý server, ktorý chce používať HTTPS protokol si musí zriadiť digitálny certifikát s verejným kľúčom. Tento verejný kľúč je použitý pri TLS Handshake na vyjednanie parametrov HTTPS spojenia. Následne je celá HTTPS komunikácia šifrovaná pomocou vyjednaného kľúča. Nieje potreba žiadnym spôsobom meniť HTTP aplikácie kvôli použitiu HTTPS, keďže o zabezpečenie sa stará protokol SSL/TLS, ktorý nezasahuje do aplikačnej vrstvy. V dnešnej dobe podporuje HTTPS veľký počet webových aplikácií. Môžu to byť aplikácie, ktoré vlastnia banky, internetové obchody, sociálne siete.

HTTPS používa port 443 čím sa líši od HTTP, ktoré používa port 80.[15]

2.1.5 HTTP Strict Transport Security (HSTS)

HSTS je mechanizmus, ktorý sa používa na vynútenie HTTPS komunikácie s webovou stránkou a zakázanie HTTP komunikácie. Jeho úlohou je zaistiť komunikáciu s web stránkou vždy iba cez zabezpečený protokol. HTTP hostiteľ môže sám seba vyhlásiť za HSTS hostiteľa poslaním Strict-Transport-Security HTTP hlavičky klientskej aplikácii cez bezpečné pripojenie (napr. TLS). Klientská aplikácia si uloží tohoto hostiteľa medzi známych HSTS hostiteľov, ak prijatie a spracovanie hlavičky prebehne správne.

HSTS politika nariaďuje klientským aplikáciam komunikovať so známymi HSTS hostiteľmi iba pomocou zabezpečeného pripojenia a špecifikuje zásady uchovania politiky a dobu trvania politiky.

HSTS politika je explicitne nadriadená nad správanie klientskej aplikácie pri preklade jednotného identifikátora zdroja (URI), pri požívateľských vstupoch (pomocou zadania adresy web hostiteľa do prehliadaču) alebo pri vložení akejkoľvek inej informácie, ktorá by pri absencii HSTS politiky prinútila klientské aplikácie komunikovať pomocou nezabezpečeného protokolu.

HSTS politika môže obsahovať voliteľnú smernicu includeSubDomains, ktorá špecifikuje že táto HSTS politika sa vzťahuje na všetkých hostiteľov ktorých doménové meno je subdoménou domény známeho HSTS hostiteľa. HSTS hlavička musí obsahovať povinnú smernicu max-age, ktorá špecifikuje čas v sekundách od prijatia HSTS hlavičky, počas ktorého platí HSTS politika. [11] Táto politika je obrovský nepriateľ tejto práce, keďže zakáže použitie nedôveryhodného certifikátu. V budúcnosti sa pokúsime zaistiť aby sme dokázali vykonať MitM útok aj na webové lokality, ktoré používajú HSTS protokol.

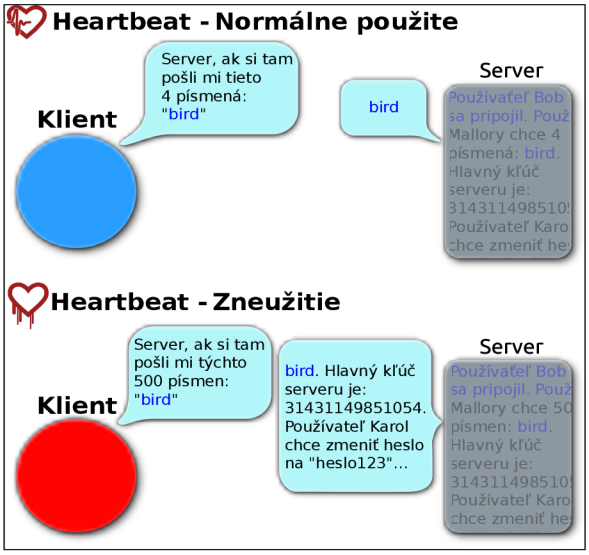
2.2 Známe útoky a chyby

Od začiatku existencie SSL/TLS ubehlo veľa času a tým sa aj odhalilo veľa rôznych chýb a zraniteľností v tomto protokole. Popísaných je len niekoľko najzaujímavejších útokov, ktoré súvisia s touto témou.

2.2.1 Hearthbleed

V marci 2014 výskumníci objavili veľmi nebezpečnú chybu v široko používanej knižnici OpenSSL. OpenSSL je kryptografická knižnica, ktorá sa používa pre zabezpečenie pripojenia v populárnych serverových aplikáciach napr. Apache a Ngnix. Heartbleed[3] umožní útočníkovi čítať citlivé dáta z pamäte zraniteľných serverov. Táto pamäť môže obsahovať

[3] http://heartbleed.com/



Obr. 2.1: Heartbleed

rôzne kryptografické kľúče, prihlasovacie údaje a mnoho ďalších diskrétnych informácií a citlivých dát.

V prípade, že by sa útočníkovi podarilo získať privátny kľúč servera, bolo by možné použiť tento kľúč pre MitM útok popisovaný v tejto práci, bez žiadneho upozornenie klientskou aplikáciu, v prípade že kľúč bol podpísaný dôveryhodnou CA autoritou. Taktiež by bolo možné pomocou takého kompromitovaného kľúča podpísať množstvo ďalších certifikátov, ktorým by klientské aplikácie dôverovali. Avšak väčšina serverov používa opravenú verziu OpenSSL knižnice.

Táto chyba je považovaná za jednu z najhorších, ktoré sa týkajú SSL/TLS. Avšak chyba nieje kryptografické zlyhanie, ale zlyhanie v implementácii Heartbeat rozšírenia SSL/TLS protokolu v OpenSSL knižnici. SSL Heartbeat je rozšírenie protokolu SSL/TLS, ktoré má za úlohu udržovať spojenie medzi serverom a klientom aby, nebolo treba znova vykonávať vyjednávací proces Handshake, ktorý je relatívne časovo náročný. Toto udržiavanie spojenia je vykonávané pomocou posielania správ serveru, ktorý tieto správy posiela naspäť klientovi čím sa udrží spojenie. Chyba spočíva v tom, že keď klient pošle správu serveru a hovorí, že dĺžka tejto správy je iná (väčšia) ako v skutočnosti je, server neskontroluje skutočnú, dĺžku správy ale použije klientom špecifikovanú dĺžku správy. Takto server odošle klientovi správu dlhšiu ako mu klient naozaj odoslal, čím sa klient dostane k pamäti serveru, ktorá mala byť bezpečne skrytá. Tento proces môžeme vidieť na obrázku 2.1. Maximálna veľkosť takejto správy môže byť 216 bajtov.[12]

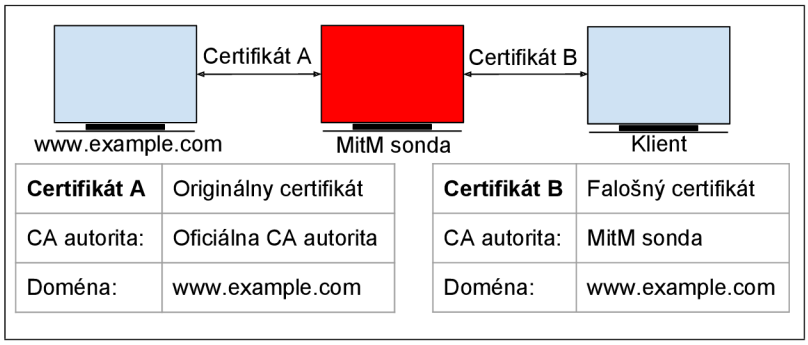
2.2.2 Sslstripping

Sslstripping je útok ktorý funguje ako SSL/TLS proxy. Počas tohto útoku útočník odstráni použitie SSL/TLS protokolu v komunikácii medzi sondou a klientom. Odstránenie SSL/TLS zapríčiní iba HTTP komunikáciu, ktorá nepoužíva žiadne zabezpečenie a je možné ju jednoducho odpočúvať a modifikovať útočníkom. Tento útok sa dá zaradiť medzi downgrade útoky. V kontexte webovej prevádzky je tento typ útoku možný iba v prípade že klient na začiatku komunikácie navštívi webovú aplikáciu prostredníctvom HTTP. Obyčajne sa tomuto útoku dá predísť použitím HSTS politiky. [10] Pre vykonanie tohoto útoku je nutné

mať prístup k sieťovej prevádzke medzi odpočúvanými stranami. V kapitole 5 bude bližšie popísané ako je možné získať prístup k takejto sieťovej premávke.

2.2.3 SSLsplitting

Tento typ útoku tiež používa SSL/TLS proxy ako predošlí spomínaný útok. Takáto proxy čaká na pokus o nadviazanie komunikácie medzi odpočúvanými stranami. Táto proxy spracováva všetku sieťovú prevádzku, ktorú si klient a server posielajú. Proxy ukončí pripojenie klienta so serverom a nadviaže vlastné pripojenie so serverom, kde sa tvári ako komunikujúci klient a s klientom nadviaže ďalšie spojenie, v ktorom sa tvári ako server. Rozdiel s predošlým typom útoku je že v tomto prípade je komunikácia medzi klientom a serverom šifrovaná. Pre šifráciu sa použije certifikát, ktorý vlastní proxy. Takéto pripojenie je ilustrované na obrázku 2.2.



Obr. 2.2: sslsplitting

2.2.4 Man in the Middle (MitM) útok

MitM je útok, pri ktorom sa útočník neprávom votrie do komunikácie medzi dvoma stranami v sieti za účelom odpočúvať a falšovať informácie, ktoré si navzájom posielajú po sieti. Tieto dve strany si zatiaľ myslia že komunikujú priamo jedena s druhou. Názov Man in the Middle je odvodený z basketbalového scenáru, kde sa dvaja hráči snažia prihrať si navzájom loptu a tretí hráč sa snaží im loptu uchmatnúť.

Tento útok spočíva v tom, že útočník je pripojený v sieti medzi klientom a serverom. Keďže komunikácia prúdi cez útočníkove zariadenie, útočník môže rôzne pracovať s touto sieťovou prevádzkou. MitM útok je veľmi jednoduché vykonať, v prípade, že má útočník prístup k pripojeniu do siete klienta alebo k pripojeniu medzi sieťou klienta a Internetom. Pakety prúdiace cez zariadenie útočníka je možné pozmeniť, zaznamenávať, zmazať a podobne.

Brániť sa proti MitM útoku je možné pomocou zašifrovania posielaných informácií po sieti. Takýmto spôsobom nebude útočník môcť zistiť aké informácie prúdia po sieti, preto že by potreboval kľúč na dešifrovanie týchto informácií. Avšak k takémuto kľúču majú prístup len komunikujúce strany. Takúto šifráciu poskytuje napríklad SSL/TLS protokol alebo SSH protokol. Tento typ útok bude popísaný podrobnejšie v kapitole 3.

2.2.5 Superfish

Superfish zachytáva HTTPS prevádzku pomocou CA certifikátu, ktorý je podpísaný sám sebou a importovaný do úložného priestoru certifikátov vo Windows. S použitím tohoto certifikátu superfish vygeneruje svoje vlastné certifikáty, ktoré sú potom použité pri HTTPS komunikácii s web hostiteľmi, bez toho aby používateľ vedel že verejný kľúč, ktorý je použitý na šifráciu ich komunikácie nepatrí ozajstnému vlastníkovi web stránky. Týmto spôsobom superfish dokáže modifikovať navštevované web stránky napr. pridaním reklamných elemnetov a podobne. Toto správanie môžeme nazvať Man in the Middle útok. Lenovo v roku 2014 tento software predinštalovávalo na niektorých notebookoch s OS Windows. [13]

2.3 Ďalšie šifrovacie protokoly

V súčastnosti je mnoho rôznych šifrovacích protokolov, aby bolo možné šifrovať a zaistiť súkromie pri komunikácii s rôznymi aplikáciami a servermi.

Secure Shell Secure Shell (SSH) používa public-key kryptografiu na overenie vzdialeného počítaču a na umožnenie autentifikácie používateľa. SSH je možné použiť rôznymi spôsobmi napríklad použiť automaticky vygenerované páry verejné a privátne kľúče pre šifrovanie sieťového pripojenia a autentizáciu používateľa pre jeho prihlásenie.

Ďalší spôsob je manuálne vygenerovať dvojicu privátny a verejný kľúč pre vykonanie autentifikácie bez použitia hesla. V tomto scenáre je nutné verejný kľúč uložiť na všetkých zariadeniach, ku ktorým sa chceme pripojiť. Táto autentifikácia je založená len na overení pomocou privátneho a verejného kľúča. Privátny kľúč nieje nikdy posielaný po sieti počas autentifikácie. SSH iba overí či osoba, ktorá ponúka verejný kľúč tiež vlastní k nemu patriaci privátny kľúč.

Secure Shell verzia 2 Tent protokol je novšia verzia SSH, ktorá používa *DHE* algoritmu na výmenu kľúčov. Verzia 2 je bezpečnejšia, efektívnejšia a prenosnejšia oproti jej predchodcovi. Taktiež obsahuje SFTP protokol, ktorého funkcionalita je podobná ako FTP, avšak je šiforvaný pomocou SSH2. Ponúka podporu viacero typov verejných kľúčov. V dnešnej dobe je používaná hlavne táto verzia protokolu.

Tieto protokoly je taktiež možné odpočúvať pomocou MitM útoku. V súčastnosti sú dostupné nástroje, ktoré dokážu prinútiť použitie falošného kľúča pre šifráciu SSH komunikácie.[2]

Kapitola 3

Analýza MitM nástrojov

K vykonaniu MitM útoku je potrebné presmerovať sieťovú komunikáciu do útočníkového zariadenia a následne naspäť do siete. Takéto presmerovanie je možné dosiahnuť niekoľkými spôsobmi na rôznych vrstvách OSI modelu. Keď komunikácia prúdi cez útočné zariadenie je možné zachytávať pakety a upravovať tieto pakety bez povšimnutia komunikujúcich zariadení. V prípade zachytenia HTTPS paketov útočník nevie čo znamenajú keďže sú šifrované pomocou SSL/TLS. Pre to je nutné použiť ďalšie nástroje, ktoré nám umožnia dešifrovať šifrovanú komunikáciu. Niektoré z týchto nástrojov si popíšeme v tejto kapitole.

3.1 SSLsplit

SSLsplit[1] je nástroj pre MitM útoky na dešifráciu SSL/TLS komunikácie. Sieťová komunikácia je presmerovaná do SSLsplit, ktorý zruší SSL/TLS pripojenie a vytvorí nové s originálnou destináciou a zapisuje všetky vyslané dáta. Toto nové pripojenie je vytvorené pomocou certifikátu, ktorý bol vložený do SSLsplit pomocou parametrov pri spustení.

Pomocou parametrov je možné použiť CA certifikát, ktorý SSLsplit použije na podpísanie certifikátov, vygenerovaných SSLsplit-om pre navštevovanú destináciu. SSLsplit je generické SSL/TLS proxy. Dokáže vykonávať MitM na všetkých druhoch bezpečného SSL/TLS pripojenia. Tento nástroj sa správa ako zariadenie medzi klientom a serverom.

SSLsplit vezme pripojenie a predstiera že je server, na ktorý sa klient snaží pripojiť a zároveň predstiera že je klient, s ktorým server komunikuje. Aby sa mu toto podarilo, generuje certifikát pre každý navštívený server. Tento certifikát je podpísaný CA certifikátom. Klient musí tomuto CA certifikátu dôverovať alebo si ho naimportovať inak dostane upozornenie o nedôveryhodnom certifikáte.

Napr. v prípade keď klient chce poslať email cez zabezpečený smtp server (napr smtp.gmial.com port 465) sslsplit vytvorí certifikát pre *smtp.gmail.com* a klientovi predstiera že je *Gmail* poštový server. Následne v upstream sa SSlSplit pripojí na server ako obyčajný klient a preposiela všetku aktivitu, ktorú klient vykonáva a odpovede serveru na danú aktivitu preposiela klientovi. Medzi tým SSLsplit má prístup ku všetkým spracovávaným dátam.

Nevýhodou je, že má minimálnu podporu prekonania HSTS mechanizmu. To znamená, že ak sa pokúsime pripojiť na webovú stránku s HSTS po prvý krát, podarí sa nám akceptovať falošný certifikát. Toto je dosiahnuté tým, že SSLsplit odstráni všetky HTTP hlavičky týkajúce HSTS. Bohužiaľ pre webové lokality, ktoré sú napevno pridané do zoznamu HSTS stránok v prehliadači, nieje možné obísť mechanizmus HSTS.

[1] https://www.roe.ch/SSLsplit

SSLsplit podporuje IPv4 a IPv6 protokoly. Pre správnu funkčnosť IPv6 je nutné použiť NAT stroj, ktorý taktiež podporuje IPv6. Takéto nástroje sú napríklad pf, ipfw (možné použiť na FreeBSD, OpenBSD a MacOS X) a tproxy (použiteľné iba na Linux). Pomocou daného NAT stroju je treba presmerovávať pakety na vstupné porty SSLsplitu.

Tento nástroj nepodporoval ukladanie hlavných hesiel (master secret) SSL/TLS relácií. V tejto práci bola táto funkcionalita doplnená. Pomocou týchto hesiel je možné dešifrovať všetky zachytené pakety bez ohladu na to aký bol použitý mechanizmus pre výmenu týchto kľúčov. Týmto spôsobom je možné dešifrovať aj relácie, ktoré používajú ECDHE alebo DHE pre výmenu kľúčov.

Výstup komunikácie zachytávanej sú súbory *.log. Tieto súbory sú v textovom formáte a obsahujú jednotlivé dátové prúdy (streams of data, tcp flows), ktoré sú zaznamenávané z 5-7 vrstvy OSI. Tieto prúdy sú už dešifrované. Toto zaznamenávanie je možné vypnúť, aby sa znížila práca s diskom.

Samotný SSLsplit nedokáže zachytávať sieťovú prevádzku do formátu pcap, no existuje niekoľko iných možností ako získať dešifrované dáta v tomto formáte. Vo vývojárskej verzii SSLsplit existuje skript, ktorý dokáže získané *.log súbory preformátovať na pcap formát. Avšak, týmto spôsobom sa stratia niektoré informácie, ktoré sa dajú získať iba pomocou záchytu na 2 a 3 OSI vrstve.

V súčasnej dobe preferujú prehliadače ECDHE a DHE výmenu kľúčov. Toto spôsobí nemožnosť rozšifrovať zachytených SSL/TLS relácií pomocou privátneho kľúča serveru. SSLsplit umožňuje špecifikovať podporované šifrovacie sady, ktoré sú ponúknuté klientovi. Takto je možné klientovi ponúknuť iba sady, ktoré používajú RSA výmenný mechanizmus.

V najnovšej vývojárskej verzii, je možné ukladať vygenerované certifikáty do špecifikovanej zložky. Potom je možné tieto certifikáty prekonvertovať na *.pem a používať stále tie isté certifikáty pre navštívené domény. Po reštarte SSLsplitu nebude klient musieť znova akceptovať nové certifikáty a nedostane upozornenie o neznámom certifikáte.

3.2 Mitmproxy

Interaktívny konzolový program, ktorý umožňuje odpočúvať, analyzovať a modifikovať sieťovú premávku pomocou Man in the Middle proxy. Na vstup je možné taktiež vložiť CA certifikát alebo konkrétny certifikát pre konkrétny server. Tento nástroj dokáže vykonávať MITM útok iba na HTTP a HTTPS protokoly. Ako názov naznačuje je tento nástroj typu proxy. Používa sa na penetračné testovanie aplikácií komunikujúcich pomocou SSL/TLS. Pomocou tohoto nástroju je možné donútiť posielať aj dáta, ktoré sú už v cache klientskej aplikácie, tieto dáta by chýbali pri pokuse o rekonštrukciu webovej komunikácie. Tento nástroj ponúka 4 operačné módy. Vyhovujúci mód je nutné vybrať podľa toho aký máme prístup ku klientskému zariadeniu a zariadeniu servera.

Veľká výhoda tohoto nástroja je, že po nastavení cesty k súboru v premennej *SSLKEYLOGFILE* sa do neho uložia všetky hlavné heslá (master secret) SSL/TLS relácií. Pomocou uložených hesiel môžeme dešifrovať zachytené relácie, bez ohľadu na použitý mechanizmus výmeny hlavných hesiel. Toto je výhoda, keďže v dnešnej dobe začali prehliadače preferovať ECDHE a DHE mechanizmus, ktorý sa nedá rozšifrovať pomocou privátneho kľúča serveru. Ďalšou možnosťou ako zariadiť aby boli dáta rozšifrovateľné je vynútenie použitia RSA na výmenu kľúčov. Mitmproxy práve toto umožňuje.

Tento nástroj dokáže ukladať premávku v textovom formáte. V uloženom súbore sa nachádzajú dešifrované dáta, a taktiež použitý certifikát pre danú webovú stránku. Tieto dáta sú získané z 5-7 OSI vrstiev, rovnako ako v SSLsplit. Pomocou tohoto nástroja nieje

možné ukladať dáta do pcap formátu. Napriek tomu, je možné zachytiť šifrované pakety inými nástrojmi na 2 a 3 OSI vrstve napr pomocou wireshark a rozšifrovať dáta pomocou privátneho kľúča servera alebo pomocou vyššie spomínaného súboru s hlavnými heslami.

Bohužiaľ mitmproxy nedokáže nijakým spôsobom obísť HSTS pri použití certifikátu, ktorý je vydaný nedôveryhodnou certifikačnou autoritou. Je to spôsobené tým, že je nástroj určený pre prostredie, v ktorom je naimportovaný používaný CA certifikát.

V prípade, že používateľ nepotrebuje GUI a chce iba zaznamenávať prevádzku, je možné použiť príkaz mitmdump. Je to nástroj pre príkazový riadok. Toto je výhoda pre automatizáciu, keďže program pobeží rýchlejšie a nieje nutné obsluhovať žiadne GUI pre zaznamenávanie dát.

3.3 Man in the Middle framework

MITMf [2] je framework, ktorý obsahuje množstvo nástrojov pre automatizáciu Mitm útoku. Pre nás je najzaujímavejší nástroj sslstrp+, ktorý umožní prekonanie HSTS. Ďalšie zaujímavé nástroje sú arpspoof, dnsspoof. Tento framework automaticky presmeruje všetku sieťovú komunikáciu na svoj vstup a vykoná MitM útok, podla špecifikovaných parametrov pri spustení. Používa rôzne knižnice, ktoré umožňujú rôzne možnosti použitia. Dokáže presmerovávať sieťovú prevádzku pomocou falšovania ARP, ICMP, DHCP alebo DNS. Výhodou MITMf je dobrá podpora HSTS

Tento framework má v sebe zabudované SMB, HTTP a DNS servery, ktoré môžu byť kontrolované a použité rôznymi modulmi. Tiež obsahuje modifikovanú verziu SSLStrip proxy, ktorá umožňuje modifikáciu HTTP a čiastočné obídenie HSTS. [1]

Tento framework používa *SSLstrip+*[3] a špeciálny DNS server *dns2proxy*[4] pre implementáciu čiastočného obídenia HSTS. Hlavný rozdiel je v použitom SSL/TLS proxy v tomto nástroji. Pripojenia používajúce SSL/TLS protokol sú tiež ukončené touto proxy, avšak, komunikácia medzi klientom a útočníkom nepoužíva SSL/TLS šifráciu a zostáva dešifrovaná. Čiastočné obídenie HSTS je urobené pomocou presmerovania klienta z navštevovaného webového hostiteľa na nepravú doménu pomocou posielania HTTP presmerovacích odpovedí. Klient je potom presmerovaný na doménové meno s extra *w* vo *www* alebo s predponou *web.* napríklad *web.example.com*. Tímto spôsobom webový hostiteľ nieje považovaný ako člen interného HSTS zoznamu webového prehliadača a klient môže pristúpiť na web stránku bez SSL/TLS pripojenia. Nepravé doménové mená sú neskôr preložené na správne IP adresy pomocou špeciálneho DNS serveru, ktorý ráta so spomínanými zmenami v doménových menách. Veľkou nevýhodou tohoto scenáru je, že klient musí nadväzovať pripojenie pomocou HTTP protokolu, aby bolo možné vykonať HTTP presmerovanie. V prípade, že sa klient pokúsi priamo nadviazať spojenie so serverom cez HTTPS pripojenie, nieje možné vykonať potrebné presmerovanie.

3.4 Porovnanie nástrojov

Hlavnou nevýhodou *SSLsplit* a *mitmproxy* je chýbajúca podpora prekonania HSTS pre hostiteľov prítomných v predinštalovaných HSTS zoznamoch webových prehliadačov, v porovnaní s *MITMf*, ktorý toto HSTS prekonanie podporuje a umožní klientom pripojenie

[2] https://github.com/byt3bl33d3r/MITMf
[3] https://github.com/byt3bl33d3r/sslstrip2
[4] https://github.com/LeonardoNve/dns2proxy

s takýmito webovými hostiteľmi.

	SSLsplit	Mitmproxy	MITMf	Bettercap
IPv4	✔	✔	✔	✔
IPv6	✔	✘	✔	✔
ukladanie kľúčov relácií	✘	✔	✘	✘
HSTS prekonanie	✘	✘	✔	✔
SSLsplitting	✔	✔	✘	✘
SSLstripping	✘	✘	✔	✔
Arpspoof	✘	✘	✔	✔
Transparentné proxy	✘	✘	✘	✘
CA certifikáty	✘	✔	✘	✘
Pcap výstup	✘	✘	✘	✔
Auto presmerovanie	✘	✘	✔	✔

Obr. 3.1: Tabuľka znázorňujúca možnosti jednotlivých MitM nástrojov.

Žiaden zo spomenutých nástrojov nedokáže zaznamenávať sieťovú prevádzku z 2. *ISO-/OSI* vrsvty do pcap formátu, keďže majú prístup a pracujú len so 7. aplikačnou vrstvou *ISO/OSI* sieťového modelu. Avšak, táto slabosť môže byť eliminovaná pomocou externého nástroja pre zaznamenávanie paketov z 2. vrstvy do pcap súboru. Tieto dáta však nebudú dešifrované. Dešifráciu je nutné vykonať pomocou privátneho kľúča, ktorý patrí k použitému certifikátu. Dešifrácia pomocou privátneho kľúča nieje možná pri použití DH algoritmu na výmenu hlavných kľúčov relácií. Avšak, *mitmproxy* dokáže hlavné kľúče relácií ukladať do súboru s NSS Key Log Formátom[5]. Pri použití tohoto súboru pre dešifráciu paketov z pacp súboru nevzniká problém pri dešifrácii paketov z relácií, pri ktorých bola použitá DH výmena kľúčov. Na obrázku 3.1 je možné vidieť porovnanie schopností spomínaných nástrojov.

[5]https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS/Key_Log_Format

Kapitola 4

Návrh automatizácie

Preto aby sme dokázali vykonať MitM útok a zachytiť sieťovú komunikáciu, v ktorej si klient a server vymieňajú informácie, je nutné aby sonda mala prístup k všetkej sieťovej komunikácii. Avšak aby sme dokázali zabezpečiť dešifráciu zachytených dát je nutné aby všetka sieťová komunikácia prúdila cez sondu v reálnom čase. V závislosti na tom akým spôsobom je sonda zapojená do siete, je nutné zvoliť správny nástroj, ktorý umožní presmerovať všetku sieťovú prevádzku cez sondu v reálnom čase. Tieto nástroje sa nazývajú *arpspoof* a *bridge*. V kapitole 5.2 bude presnejšie popísané použitie týchto nástrojov a scenáre ako sondu zapojiť do siete podľa použitých nástrojov.

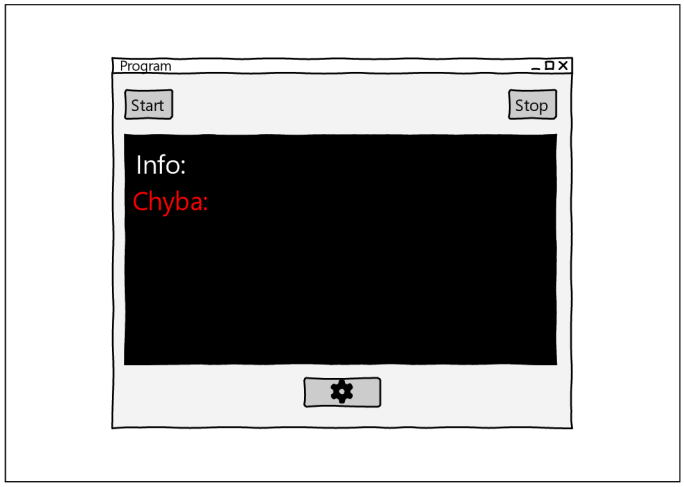
Po presmerovaní komunikácie cez sondu, je nutné celú komunikáciu ukladať na disk, aby bolo možné použité dáta analyzovať v budúcnosti rôznymi forenznými nástrojmi. Väčšina forenzných nástrojov nepodporuje vstup z aplikačnej vrstvy ale podporuje vstup zo súboru v *pcap* formáte. Do tohoto formátu nedokáže zaznamenávať dáta žiaden z analyzovaných MitM nástrojov. Preto je nutné siahnuť po alternatívnom riešení. Použitím Linuxových aplikácií, je možné zachytiť sieťovú komunikáciu z 2. *OSI* vrstvy do *pcap* súboru. Pre toto zachytávanie dát som sa rozhodol použiť *tcpdump*, čo je voľne dostupný Linuxový nástroj pre to určený.

Zachytená komunikácia je v prípade šifrovaného pripojenia pomocou SSL/TLS pre útočníka bezcenná, pokiaľ nemá prístup k potrebným kľúčom na dešifráciu komunikácie. Preto aby bolo možné dešifrovať zachytenú komunikáciu, je nutné použiť SSL/TLS proxy, ktoré nám umožní prístup k nevyhnutným kľúčom pre dešifráciu. Takéto proxy ponúkajú nástroje popísané v kapitole 3. Po analýze a zistení čo, ktorý nástroj dokáže bol zvolený nástroj SSLsplit. Tento nástroj bol zvolený hlavne kvôli jeho rýchlosti a podpore IPv6, s čím mali ostatné nástroje problém.

Aby bolo možné použiť spomínané SSL/TLS proxy, je nutné aby prichádzajúce a odchádzajúce pakety boli presmerované na vstup použitého proxy aby toto proxy mohlo spracovať dané pripojenia. Toto presmerovanie nám umožní *firewall*, ktorý dokáže presmerovať premávku z jedného portu na iný. V použitom OS je dostupný nástroj *iptables*, ktorého použitie bude bližšie popísané v kapitole 5.2.3.

Automatizácia bude vytvorená pomocou spomínaných Mitm nástrojov a python skriptu, ktorý zabezpečí správne nastavenie vyššie spomínaných potrieb. Aby bolo preposielanie čo najrýchlejšie bude predvolené použitie mostu medzi 2 rozhraniami, ktoré sú dostupné na použitom hardvéry. Tento most bude taktiež automaticky vytvorený pri spustení automatického MITM. Skript bude možné ovládať pomocou parametrov v príkazovom riadku a pomocou GUI.

V neposlednom rade bude vytvorené grafické používateľské rozhranie (GUI). V GUI



Obr. 4.1: Návrh hlavného okna GUI

aplikácie bude možné jednoducho upraviť nastavenia sondy. V GUI aplikácie bude možné vidieť či sa správne spustil útok a v prípade neúspešného štartu budú na výstup uvedené chyby, ktoré znemožnili spustenie útoku. Na obrázku 4.1 je možné vidieť návrh hlavného okna v GUI aplikácii. Je tu vidieť tlačítka pre spustenie útoku, ukončenie útoku, nastavenie sondy a textové pole s informáciami o súčasnom stave útoku. Ďalej bude vytvorené okno s jednoduchým nastavením útoku a sondy. V tomto okne bude možné vybrať typ útoku.

Výber HW

Na hardware sondy bolo niekoľko nárokov. Nutné bolo aby sonda mala dva gigabitové Ethernet vstupy pomocou, ktorý sa vytvorí most medzi týmito dvoma vstupmi. SSD disk, na ktorý sa budú ukladať zachytávané dáta. Ďalej bolo nutné zvoliť procesor a RAM aby dokázali zachytávať prenášané dáta. Po preskúmaní možností bola vybraná konfigurácia (Obrázok 4.2).

	Meno výrobku	
Procesor	Intel Pentium G3258	2 jadrá - 3.2GHz
Základná doska	GIGABYTE GA-Z97N-WIFI	2x GLAN
Disk	Kingston HyperX FURY	120GB
RAM	Crucial Ballistix Sport	8GB
Skriňa	Chieftec Compact Series IX-03B-OP	197 x 63 x 220 mm
Napájanie	Chieftec CDP-090ITX	7.5A

Obr. 4.2: Tabuľka so zvoleným hárdverom.

Na základe niekoľko rád sme sa rozhodli pre značku Intel sieťových kariet, z dôvodu väčšej priepustnosti. SSD disk sme zvolili o veľkosti 120GB, ktorý bude mať na laboratórne účely dostatočnú kapacitu. RAM s kapacitou 8GB, intel procesor s frekvenciou 3,2GHZ a základnú dosku GIGABYTE GA-Z97N-WIFI - Intel Z97 ktorá má zabudované dve gigabitové sieťové karty (Intel a Realtek).

Výber OS

Ako operačný systém bol zvolený Archlinux, keďže pre použitie niektorých nástrojov pre realizáciu MitM útoku je nutný Linuxový systém. Archlinux bol zvolený práve pre to, že nám ponúka jednoduchú inštaláciu potrebných nástrojov, ktoré by bolo nutné kompilovať zo zdrojových kódov na väčšine iných operačných systémoch. Toto je zabezpečené pomocou repozitárov, ktoré sú spravované komunitou.

Kapitola 5

Implementácia

Automatické spustenie MitM útoku poskytuje python skript. Skript podľa vstupných parametrov postupne spustí konfiguráciu sondy a nástroje potrebné pre útok. Skript je možné spustiť tiež s grafickým užívateľským rozhraním.

Pred spustením automatického skriptu pre MitM je nutné nainštalovať všetky potrebné nástroje, ktoré skript používa a pripraviť operačný systém, na ktorom bude skript bežať. Ako operačný systém bol zvolený Archlinux. Pre jeho správnu inštaláciu bolo nutné naformátovať, správne rozdeliť miesto používaného USB a vytvoriť súborové systémy na vytvorených partíciách. Taktiež bolo nutné pripraviť súborový systém na SSD disku, ktorý bol použitý pre rýchle zaznamenávanie sieťovej prevádzky.

Na sondu bolo potom možné nainštalovať zvolený Archlinux a potrebné nástroje napr. *bridge-utils*, *arpspoof*, *tcpdump* a *openSSL*. Použitý nástroj SSLsplit bolo nutné stiahnuť z git repozitára , v ktorom sa nachádza upravená verzia, ktorá bola vylepšená o zaznamenávanie hlavných kľúčov SSL/TLS relácií. V stiahnutej zložke je tiež nutné skompilovať stiahnuté zdrojové kódy.

5.1 Generácia CA kľúčov

Aby bolo možné použiť MitM nástroje, je nutné mať CA certifikát, ktorý bude použitý pre podpisovanie vygenerovaných certifikátov navštevovaných webových serverov. Pre dešifráciu tejto webovej komunikácie je nutné mať privátny kľúč patriaci k vygenerovanému CA certifikátu. Takýto CA certifikát sme vygenerovali a podpísali pomocou OpenSSL nástroja v Linuxe. Tento certifikát bolo nutné podpísať sám sebou, čím sme vytvorili koreňovú certifikačnú autoritu. V nasledujúcich kódoch je ilustrované ako je možné vytvoriť privátny kľúč (Kód 5.1). Vytvorenie požiadavku na podpis certifikátu (Kód 5.2), kde je nutné vyplniť informácie o vlastníkovi certifikátu. V neposlednom rade tu je zobrazené podpísanie certifikátu samým sebou (Kód 5.3), kde sa vygeneruje verejný kľúč.

Kód 5.1: Generácia privátneho kľúča

```
1 openssl genrsa -out ca.key 2048
```

Kód 5.2: Generácia požiadavku na podpis certifikátu

```
1 openssl req -new -key ca.key -out ca.csr
```

Kód 5.3: Podpísanie a vytvorenie verejného kľúča

```
1 openssl x509 -req -days 365 -in ca.csr -signkey ca.key -out ca.crt
```

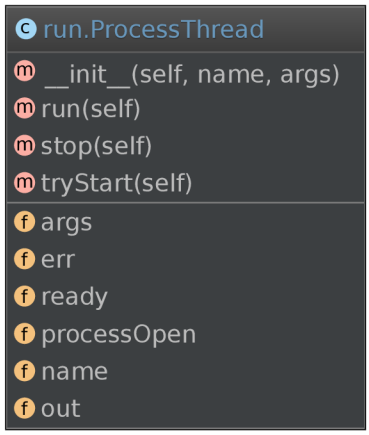
5.2 MitM Automatizér

MitM Automatizér je skript písaný v jazyku *Python*, ktorý má za úlohu spustiť presmerovanie webovej prevádzky na vstup penetračných nástrojov a spustiť potrebné nástroje pre záchyt paketov do pcap súborov. Konfigurácia zariadenia je vykonaná automaticky pri spustení skriptu, ktorý nakonfiguruje sondu a spustí MITM útok. Pri spustení je možné skript ovládať pomocou rôznych parametrov (Kód 5.4).

Kód 5.4: Nápoveda MitM Automatizéra

```
1  Mitm Automatizer
2
3  nepovinne argumenty:
4    -h, --help          Vypise tuto napovedu.
5    --arp               Pouzit arpspoof (zapojenie v~LAN).
6    --bridge            Pouzit bridge (upstream pripojenie pred LAN).
7    --nobridge          Nepouzivat bridge (uzivatel vytvoril vlastny bridge).
8    --dbg               Vypis debug informacii
9    --nopcap            NEzachytavat do pcap suboru.
10   --gui               Spustit s~GUI.
11   --ipv6              Pouzit ipv6.
12   --output FILE       Priecinok pre ulozenie zachytenych dat.
```

Používané nástroje pre automatizáciu sú spúšťané ako podprocesy MitM Automatizéru. Správu jednotlivých spustených podprocesy ponúka trieda *ProcessThread* (Obrázok 5.1), ktorá rozširuje python triedu *threading.Thread* [1]. Trieda *ProcessThread* definuje metódu *run*, ktorá zabezpečuje správne spustenie spomínaných vlákien.



Obr. 5.1: Trieda processThread

Počas spúšťania sú na štandardný výstup vypisované informačné hlášky o stave spúšťaného procesu. V že nástroj nieje nainštalovaný, vypíše sa chyba s informáciou o chýbajúcej aplikácii a program ukončí všetky spustené vlákna a vráti konfiguráciu sondy do pôvodného

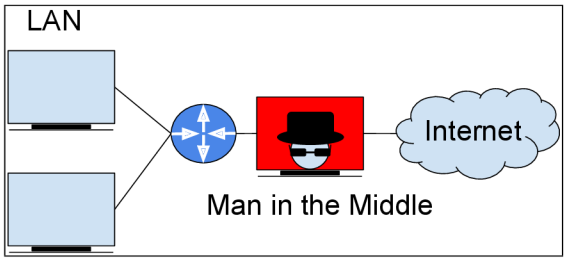
[1] https://docs.python.org/2/library/threading.html#threading.Thread

stavu. Každé vlákno je možné ukončiť pomocou metódy *stop*, ktorá ho ukončí a vypíše jeho výstup.

Pri spustení bez GUI, je možné skript ukončiť pomocou CTRL-C, čo pošle prerušovací signál. Tento signál je zachytený a následne sa spustí funkcia *cleanup* pre správne ukončenie programu. Pred ukončením programu je nutné vrátiť všetky zmeny do pôvodného stavu a zastaviť všetky spustené nástroje. Toto zabezpečí funkcia *stopThreads*, ktorá je tiež použitá pri zastavovaní útoku v GUI aplikácii a pri jej samotnom zavieraní.

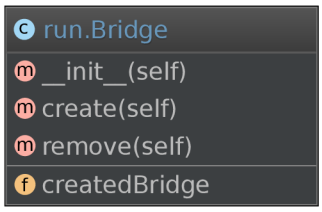
5.2.1 Most

Na presmerovanie všetkej sieťovej komunikácie je nutné sondu zapojiť z oboch strán do siete. Takéto pripojenie je možné v prípade, že máme prístup k Ethernetovému káblu pred sledovanou sieťou (Obrázok 5.2). Tento scénar je považovaný za predvolené zapojenie MitM sondy do siete. Most je vytvorený pomocou pomocou Linuxovej aplikácie *brctl*, ktorá vytvorí



Obr. 5.2: Ilustrácia zapojenia MitM sondy pred LAN

most medzi dvoma GLAN rozhraniami sondy. Ethernetový kábel s prístupom do Internetu je nutné zapojiť do rozhrania *eno1* a do druhého rozhrania *enp2s0* je nutné pripojiť Ethernetový kábel s pripojením do odpočúvanej siete. Tento most sa správa ako virtuálny sieťový switch, ktorý pracuje transparentne. Toto správanie sa nazýva *transparentný firewall*. Pri preposielaní paketov pomocou transparentného firewallu klient ani server netušia, že je medzi nimi útočník, ktorý má prístup ku všetkým paketom, ktoré si preposielajú. Týmto sa zníži schopnosť detekcie vykonávaného MitM útoku, keďže bude sonda preposielať všetku prevádzku medzi klientom a serverom na 2 vrstve *ISO/OSI*. Pomocou tohoto zapojenia má sonda prístup k všetkým paketom vyselaným medzi klientom a serverom, bez nutnosti použitia dodatočných nástrojov (napr. *arpspoof*) na presmerovanie paketov na rozhranie sondy. Pre správnu funkciu mostu je naviac nutné spustiť proces *dhclient*, ktorý umožní klientom automaticky získavať IP adresy po pripojení k sonde.



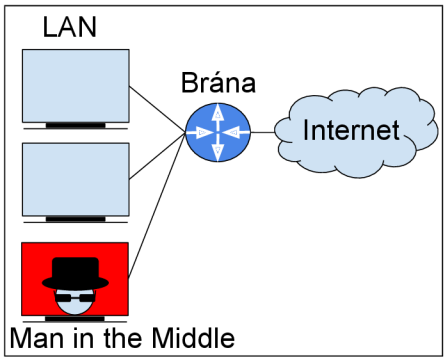
Obr. 5.3: Trieda Bridge

Pre takýto most je v skripte vytvorená trieda *Bridge* (Obrázok 5.3. Vytvorenie mostu je zabezpečené pomocou metódy *create*, ktorá postupne spúšťa príkazy nutné pre vytvorenie mostu. Pri správnom vykonaní príkazov sa do debug výstupu zapisuje výsledok spúšťaných

príkazov. Pri ukončovaní MitM útoku, sa spúšťa metóda *remove*, ktorá odstráni vytvorený most a vráti nastavenia do pôvodného stavu.

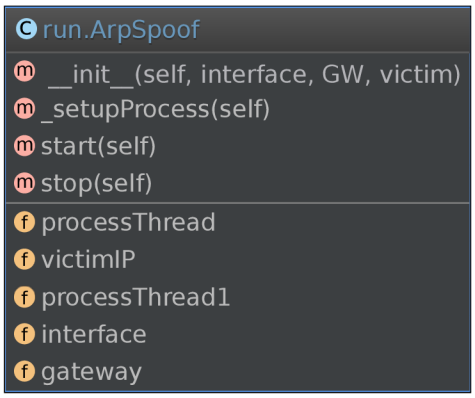
5.2.2 Arpspoof

V prípade že nieje možnosť pripojiť sondu na linku pred analyzovanú sieť, avšak máme prístup priamo do analyzovanej LAN siete (Obrázok 5.4) je nutné použiť nástroj *arpspoof*. Klient a MitM sonda sú pripojené do Internetu pomocou výstupnej brány. Výstupná brána je sieťové zariadenie, ktoré spája LAN sieť s Internetom. Toto zariadenie preposiela sieťovú premávku z lokálnej podsiete do iných pripojených podsietí. Nástroj *arpspoof* falšuje a po-



Obr. 5.4: Iluštrácia zapojenia MitM sondy v LAN

siela do siete Address Resolution Protocol (ARP) správy aby si zariadenia v sieti uložili falšované údaje do ich smerovacích tabuliek. ARP tabuľka mapuje IP adresy k fyzickým Ethernetovým (MAC) adresám. [8] IP adresa výstupnej brány je týmto spôsobom namapovaná na MAC adresu rozhrania MitM sondy, namiesto originálnej MAC adresy výstupnej brány. Tento nástroj umožňuje špecifikovať IP adresu zariadenia, ktoré bude presmerované alebo je možné presmerovať celú podsieť. Neodporúča sa pokúšať sa presmerovať celú premávku veľmi rušnej podsieti na rozhranie sieťovej karty s nedostatočnou rýchlosťou, kvôli hrozbe vysokého spomalenia siete, ba dokonca to môže viesť k odopretiu služby (DoS) v danej sieti. Tento scénar, nieje nanešťastie taký rýchly, spoľahlivý a transparentný ako vyššie spomínaný scénar s mostom.



Obr. 5.5: Trieda ArpSpoof

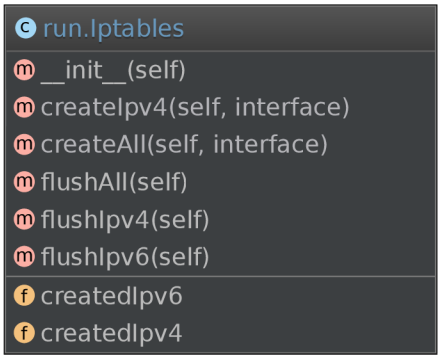
Tento nástroj je v skripte spušťaný pomocou *start* metódy v *ArpSpoof* triede (Obrázok

5.5). Táto metóda najskôr predpripravý potrebné spúšťacie argumenty pre tento program. V prípade že je nutné odpočúvať presnú IP adresu klienta, je nutné spustiť dve vlákna *arpspoof*. Jedno vlákno bude presmerovávať premávku od klienta na rozhranie sondy a druhé vlákno bude presmerovávať premávku od serveru ku klientovi. Tieto vlákna sa pri ukončovaní zastavia pomocou metódy *stop*, ktorá začne ukončovací proces.

5.2.3 Iptables

Tento nástroj slúži na konfiguráciu linxového kernel firewallu. Je používaný na analýzu, modifikáciu, preposielanie, presmerovávanie a zahadzovanie IPv4 paketov. Toto filtrovanie je umožnené pomocou kernelu a je usporiadané do kolekcie tabuliek, každá s rôznym účelom. Tieto tabuľky sú vytvorené pomocou sady preddefinovaných pravidiel. Každé z pravidiel má definované podmienky a pri splnení daných podmienok je vykonaná pridelená akcia. Týmto spôsobom je možné presmerovať webovú premávku na vstup MitM nástrojov, ktoré ju ďalej spracovávajú. Tento nástroj bol použitý na presmerovanie všetkej HTTPS (tcp port 443) a HTTP (tcp port 80) sieťovej premávky na port kde *SSLsplit* očakáva prichádzajúcu premávku. Okrem použitia *iptables* je nutné zapnúť preposielanie paketov na použitom rozhraní aby bola umožnená komunikácia medzi klientom a serverom.

V MitM skripte správu presmerovania a firewallu zabezpečuje trieda *Iptables* (Obrázok 5.6), ktorá spúšťa potrebné príkazy pre vytvorenie smerovacích tabuliek. Pri ukončovaní útouk sú tieto pravidlá zmazané pomocou metódy *flushAll* (volá metódy *flushIpv4* a *flushIpv6* v závislosti použitia IPv6 prepínača).



Obr. 5.6: Trieda Iptables

5.2.4 Tcpdump

Tento nástroj je použitý pre zachytávanie všetkej internetovej prevádzky odpočúvanej siete. Zachytené pakety sú uložené do *pcap* súboru, ktorý je možné spracovávať pomocou knižnice libpcap[2], ktorá sa použitá v nástrojoch pre analýzu sieťovej prevádzky napr. *Wireshark*. Všetky zachytené dáta budú uložené na SSD disk. Veľkosť uložených súborov je limitovaná 100MB na 1 súbor. Užívatelia si môžu navyše nastaviť limit pre maximálny počet takýchto súborov a po dosiahnutí maximálneho počtu, sa začnú tieto súbory prepisovať novými dátami od najstaršieho vytvoreného súboru. Navyše je možné nastaviť filter pre zachytávané pakety, pre zvýšenie efektivity zachytávania paketov a využitia úložného miesta. Toto je veľmi výhodné pri dlhodobom monitorovaní alebo pri nedostatku úložného miesta.

[2]http://www.tcpdump.org/manpages/pcap.3pcap.html

Tcpdump je spravovaný pomocou triedy *Tcpdump* (Obrázok 5.7. Spustenie je vykonané pomocou príkazu ktorý bol automaticky vygenerovaný metódou *_setupProcess* na základe požadovaných parametroch. Štandardne je v skripte použitý limit 1,2GB pre veľkosť jedného zachyteného súboru, čo zníži náročnosť na použitú RAM pamäť počas záchytu. Číslovanie súborov je nastavené od 0. Veľkosť súborov a max počet súborov je možné nastaviť pomocou premenných *maxFileSize* a *maxFiles*.

Obr. 5.7: Trieda Tcpdump

5.2.5 SSLsplit

Pre MitM útok bol zvolený vyššie spomínaný *SSLsplit*. Tento nástroj je písaný v programovacom jazyku C a využíva OpenSSL knižnicu tiež písanú v C, preto je nástroj veľmi rýchly. Tento nástroj podporuje všetku SSL/TLS a TCP premávku, v porovnaní s ostatnými nástrojmi.

Pomocou vyššie spomínaného nástroja *iptables* a *ip6tables* je presmerovaná premávka SSL/TLS na adresu 0.0.0.0 a port 10443 (Kód 5.5). Pre IPv6 premávku je použitý rovnaký port, no adresa na ktorej *SSLsplit* očakáva premávku je ::1, navyše je nutné použiť ako NAT stroj tproxy (Kód 5.6). Rovnaké adresy sú použité pre TCP premávku s rozdielom v použitom čísle portu 10080.

Kód 5.5: Príklad spustenia SSLsplit iba IPv4

```
1 sslsplit -D -K ca.pem -k ca.key -c ca.crt \
2     -A SSLKEYLOGFILE \
3     https 0.0.0.0 10443 http 0.0.0.0 10080
```

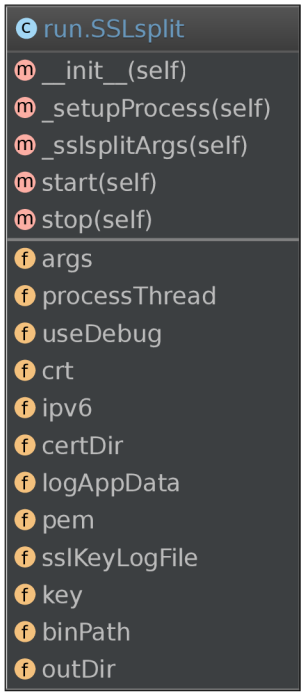
Kód 5.6: Príklad spustenia SSLsplit s podporou IPv6

```
1 sslsplit -D -e tproxy -K ca.key -k ca.key \
2     -c ca.crt -A SSLKEYLOGFILE \
3     https 0.0.0.0 10443 \
4     http 0.0.0.0 10080
```

Ako NAT stroj je pri použití IPv6 použitý *tproxy*, kvôli jeho podpore IPv4 a IPv6. Toto je jediný NAT stroj s podporou IPv6 na Linuxe, avšak na FreeBSD a OpenBSD je viacej strojov, ktoré podporujú IPv6 napríklad *pf* alebo *ipfw*. Vyššie spomínané vygenerované CA kľúče (ca.crt, ca.key, ca.pem) sú použité ako kľúče koreňovej autority pri spustení

SSLsplit. Dôležité je použiť privátny kľúč s parametrom -K (musí byť vo fromáťe *.key alebo *.pem) aby sme neskôr mohli tento kľúč použiť pri dešifrácii zachytených paketov SSL/TLS relácií, pri ktorých nebol použitý DH výmenný algoritmus. Avšak väčšina aktualizovaných moderných prehliadačov uprednostňujú práve DH algoritmus pre výmenu kľúčov.

Spracovanie, vygenerovanie správnych argumentov, spustenie a zastavenie nástroja *SSLsplit* v skripte má na starosti trieda *SSLsplit* (Obrázok 5.8). Pred štartom útoku metóda *_sslsplitArgs* spracuje požiadavky skriptu a pripraví argumenty pre spustenie nástroja.



Obr. 5.8: Trieda SSLsplit

5.2.6 SSLsplit ukladanie hlavných kľúčov SSL/TLS relácií

V prípade že použitý webový prehliadač uprednostňuje šifrovacie sady s DH algoritmom pre výmenu kľúčov môže byž použiť iný scenár pre dešifráciu zachytených paketov alebo je možné zakázať používanie šifrovacích sád s DH algoritmom pre výmenu kľúčov.

Druhý spomínaný scénar nieje možné vykonať vždy, keďže nie každý server podporuje požadované šifrovacie sady, čo môže spôsobiť nemožnosť komunikácie medzi serverom a klientom. Preto bol do *SSLsplit* doimplementovaný mechanizmus pre zaznamenávanie hlavných kľúčov SSL/TLS relácií.

Implementácia tohoto mechanizmu bola možná po lokalizovaní kde v *SSLsplit* sú spracovávané nové nadviazované pripojenia. Toto miesto bolo určené pomocou ľadiacích nástrojov použitých na *SSLsplit*. Funkcia pre spracovanie pripojení sa nazýva *pxy_bev_writecb* a je v zdrojovom súbore *pxyconn.c*. V tejto funkcii je pridané volanie novovytvorenej funkcie *pxy_sslkeylog_write()*, ktorá po nadviazaní nového pripojenia získa client_random relácie a z SSL/TLS kontextu získa dáta z premennej ssl->session->master_key. Tieto dáta sú potom prekonvertované pomocou novo vytvorenej konvertovacej funkcie ssl_data_to_hex_str do novo implementovaného zapisovača. Spustenie tohoto mechanizmu je možné pomocou

parametru -A spolu s názvom súboru, do ktorého budú kľúče ukladané. Tento mechanizmus umožňuje dešifrovať zachytené dáta bez ohľadu na použitú šifrovaciu sadu.

5.2.7 Grafické používateľské rozhranie

GUI bolo implementované pomocou Qt4 designer a PyQt4 knižnice. Je nutné ho spúšťať z používateľského účtu *root* aby malo dostatočné práva na vykonávanie zmien v konfigurácii. Pre oznamovanie stavu spustených nástrojov bol použitý PlainTextEdit widget a vytvorený špeciálná špeciálna trieda pre zaznamenávanie informácií na výstup GUI. Je to rozšírenie triedy *Handler* z python knižnice *logging* o zapisovanie správ do PlainTextEdit widgetu. V hlavnom okne GUI je možné spustiť a zastaviť útok pomocou tlačítok. Po kliknutí na tlačítko *Start* sa spustí funkcia *wrapperStart*, ktorá sa postará o zablokovanie tlačítka *Start* aby sa zabránilo pokusu o znovu spustenie útoku a pokusu o editáciu konfigurácie počas bežiaceho útoku. Následne je spustený útok zavolaním funkcie *runThreads*. Pri zatváraní alebo ukončovaní útoku je spustená funkcia *wrapperStop* a následne sú odblokované tlačidlá, ktoré boli počas útoku deaktivované. Stlačenie tlačidla s ozubeným kolesom spustí modálne okno, v ktorom je možné meniť nastavenia útoku pred samotným spustením útoku.

5.3 Dešifrácia zachytených dát

Dešifrácia obsahu webovej komunikácie šifrovanej pomocou SSL/TLS je možná pomocou dvoch bežných postupov. V niektorých prípadoch je možné zachytiť webovú komunikáciu a použiť privátny kľúč serveru pre získanie zdieľaných kľúčov relácie (záleží na použitom algoritme výmeny kľúčov), ktoré boli použité na symetrickú šifráciu danej relácie. Pomocou získaných kľúčov je možné dešifrovať zachytenú komunikáciu.

Druhá možnosť je zachytávať SSL/TLS relácie pomocou proxy. Ak máme prístup ku klientskému zariadeniu, môžete naň nainštalovať svoj vlastný CA certifikát a nakonfigurovať ho aby dôverovalo certifikátu proxy.

5.3.1 Dešifrácia pomocou privátneho kľúča

Ak máme privátny kľúč serveru, a zachytenú SSL/TLS komunikáciu, v ktorej bol použitý RSA algoritmus na výmenu kľúčov relácie, môžeme dešifrovať premaster secret odoslané klientom a pomocou neho obnoviť kľúče relácie, ktoré boli použité na symetrické šifrovanie komunikácie. Pre toto je nutné zachytiť všetky paket, ktoré obsahujú informácie o SSL/TLS handshake relácie a následnú webovú premávku medzi klientom a serverom, ktorú chceme dešifrovať. Nieje nutné mať privátny kľúč pred získavaním zachytených paketov. Je možné zachytiť sieťovú premávku a neskôr získať privátny kľúč serveru pre dešifráciu. Je nutné si uvedomiť že táto technika bude fungovať len pre SSL/TLS relácie, v ktorých bol použitý RSA algoritmus na výmenu kľúčov, ale nie pre relácie, ktoré používajú Diffie-Hellman algoritmus na výmenu kľúčov. Pre túto metódu je možné použiť nástroj Wireshark, ktorý dokáže zachytávať SSL/TLS sieťovú komunikáciu a následne pomocou privátneho kľúča serveru ju dokáže dešifrovať.

5.3.2 Dešifrácia pomocou proxy

Môžeme zachytiť a analyzovať obsah SSL/TLS komunikácie pomocou SSL/TLS proxy. Aby sa nám toto podarilo je nutné vytvoriť proxy cez ktoré bude komunikácia klienta a serveru preposielaná. Keď klient pošle dotaz na SSL/TLS server odpoveď serveru je zrušená na

proxy a proxy vytvorí šifrovaný tunel medzi proxy a serverom. Proxy podstrčí falošný certifikát servera klientovi a vytvorí SSL/TLS tunel medzi klientom a proxy. Proxy môže po tom kontrolovať a prehliadať obsah komunikácie alebo použiť zachytený obsah na ďalšiu analýzu. SSL/TLS je samozrejme vytvorené tak aby chránilo pred touto situáciou, ktorá sa nazýva MitM na SSL/TLS protokol. Webová klientská aplikácia by mala detekovať nepravý certifikát serveru, pretože tento certifikát nieje podpísaný dôveryhodnou CA a mala by varovať používateľa o tejto skutočnosti. V praxi forenzný vyšetrovatelia a autorizovaný sieťový personál môže týmto varovaniam predísť. Ak mame klientskú stanicu pod kontrolou, čo je prípad v mnoho organizáciach, môžeme naň nainštalovať lokálne vygenerované dôveryhodné CA certifikáty do každého klientského prehliadaču. Tento CA certifikát môže byť použitý touto organizáciou pre digitálne podpisovanie falošných certifikátov, ktoré poskytuje odpočúvacie proxy. Tieto certifikáty bude následne klient vidieť ako dôveryhodné certifikáty. Tiež je možné sa spoľahnúť na to, že v dnešnej dobe ešte stále mnoho používateľov ignoruje tieto bezpečnostné varovania a akceptujú falošný certifikát, čím obetujú svoju bezpečnosť a súkromie. [4]

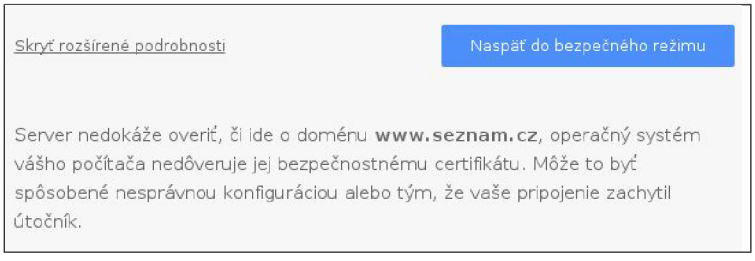
Kapitola 6

Testovanie

Základný experiment je použitie základného spomínáného scenáru so zapojením MitM sondy medzi sieť klienta a Internet (Sekcia 5.2.1) a spustiť MitM útok pomocou vytvoreného skriptu. Pri pokuse klienta o pripojenie k serveru dostane klient varovanie o nedôveryhodnom certifikáte, použitom pre SSL/TLS šifráciu (Obrázok6.1). Toto varovanie sa objaví v adresnom riadku webového prehliadača.

Obr. 6.1: Nebezpečné pripojenie

Podrobnejšie informácie budú zobrazené v prehliadači, namiesto navštevovanej webovej stránky. Toto varovanie používateľovi ponúka možnosť zobraziť pokročilé informácie a možnosti. V prípade návštevy obyčajného webového hostiteľa, ktorý nepoužíva HSTS je možné akceptovať nedôveryhodný certifikát a pridať ho medzi dôveryhodné certifikáty. (Obrázok 6.2)

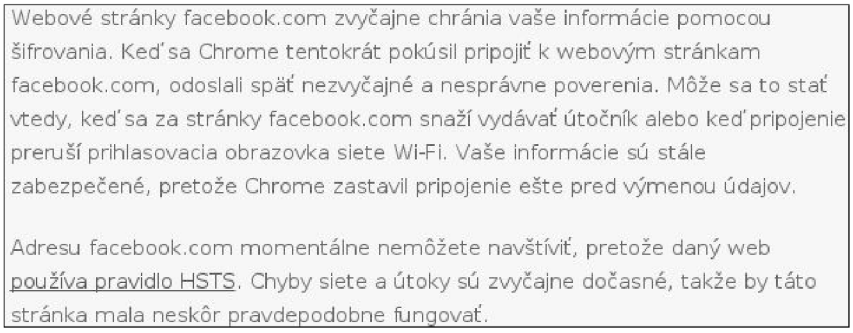


Obr. 6.2: Varovanie s možnosťou pokračovať na web.

Týmto spôsobom je možné pokračovať v pripojení na webového hostiteľa s riskovaním že

pripojenie nebude bezpečné. Toto varovanie bude prítomne pri každom pokuse o prvé nadviazanie pripojenia medzi klientom a daným serverom. Po akceptácii falošného certifikátu pre danú webovú lokalitu, už pri ďalšej návšteve lokality nebude toto varovanie zobrazené.

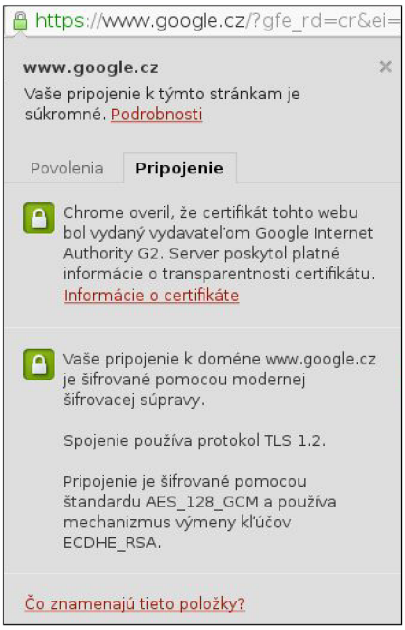
V prípade návštevy hostiteľa, ktorý sa nachádza v predinštalovanom zozname HSTS hostiteľov, webový prehliadač zobrazí podobné varovanie z predošlého experimentu, avšak naviac tu bude zobrazená informácia o tom že hostiteľ používa HSTS a nieje možné pokračovať v pripojení na danú lokalitu s nedôveryhodným certifikátom. Takéto správanie je možné vidieť pri návšteve *facebook.com* počas MitM útoku (see Figure 6.3).



Obr. 6.3: Prehliadač neponúka možnosť pokračovať na web.

Varovania o nedôveryhodnom certifikáte zmiznú po tom čo si klient naimportuje certifikát CA autority do svojeho prehliadaču alebo do svojho OS.

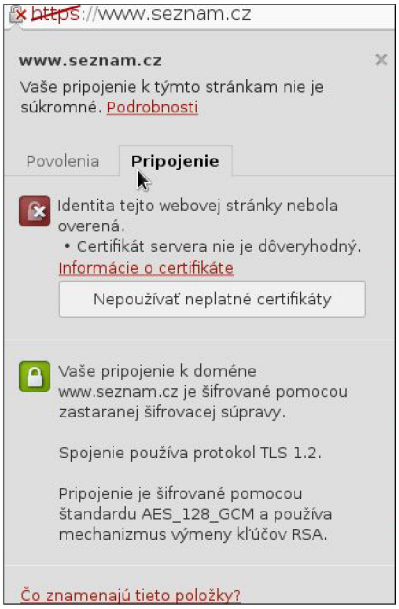
Pri navštívení webových aplikácií s dôverovaným certifikátom pomocou webového prehliadaču je znázornené bezpečné pripojenie ako napr. na obrázku 6.4. Na obrázku 6.5 vi-



Obr. 6.4: Importovaný dôveryhodný certifikát

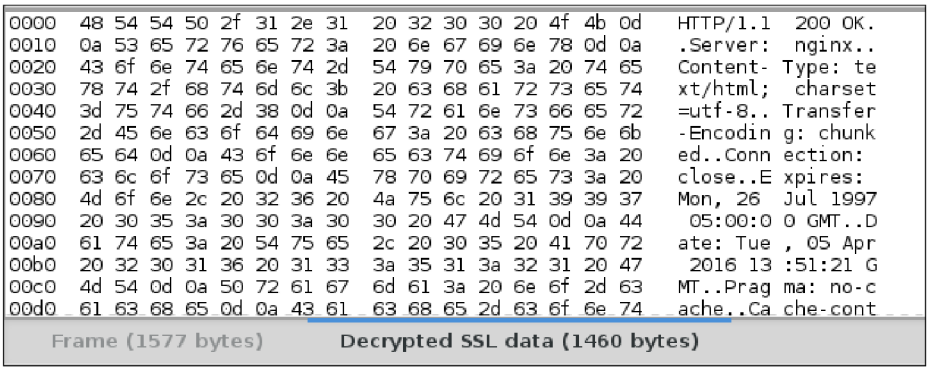
díme ako nás prehliadač upozorní pri navštívení webovej aplikácie s certifikátom, ktorý nieje v úložisku dôveryhodných certifikátov zariadenia.

Po zachytení paketov do pcap súboru a získaní prvátneho kľúča alebo získani hlavných



Obr. 6.5: Nenaimportovaný certifikát

kľúčov relácií je možné tieto pakety dešifrovať pomocou importu kľúčov do nástroja Wireshark. Importovať ich môžeme v menu Nastavenia - Protokoly - SSL. Po importe prebehne dešifrácia a je možné prehliadať dešiforvané pakety. Používateľ pre prehliadanie dešifrovaných paketov musí kliknúť na záložku *Decrypted SSL data* v označenom pakete (see Figure 6.6).

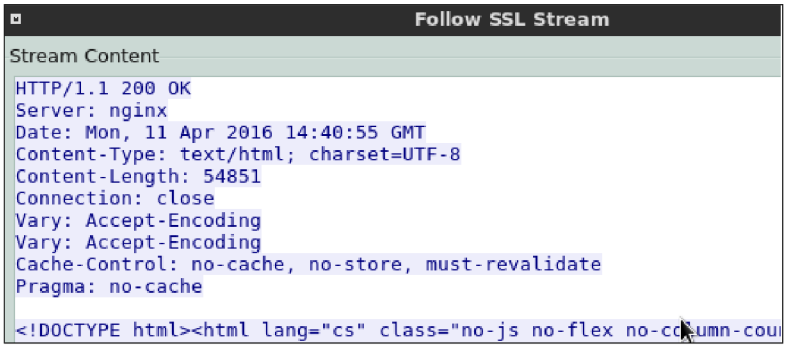


Obr. 6.6: Dešifrované dáta v Wireshark

Tieto dešifrované dáta je možné potom použiť pri podrobnejšej sieťovej analýze pomocou rôznych nástrojov napr. pre získanie chatovacích konverzácií, používateľských prihlasovacích údajov a podobne. Dáta z aplikačnej vrstvy je možno vidieť pomocou zvolenia možnosti *Follow SSL stream* (see Figure 6.7), kde je možné vidieť surové dáta HTTP protokolu.

Testovacia zostava

Zariadenia boli prepojené pomocou káblami Premiumcord UTP CAT6, ktoré podporujú gigabitový prenos. Na jednej strane sondy bol zapojený Notebook Lenovo ThinkPad T440s



Obr. 6.7: HTTP dáta vo Wireshark

(GLan, i7, ssd, 12gb ram). Tento notebook bol nastavený ako HTTPS server pomocou *Apache* a vytvorených certifikátov pomocou *OpenSSL* nástroja. Na tomto servery bol vytvorený 12GB súbor (*output.tar*), ktorý bol prenášaný po sieti medzi klientom a serverom počas testovania. MitM sonda bola nakonfigurovaná pomocou MitM Automatizéru ako most medzi klientom a serverom (viď Kapitola 5.2.1). Na druhej strane sondy bol zapojený Notebook Lenovo Ideapad G580 spolu s GLAN prevodníkom na USB3.0 a bol použitý ako klientské zariadenie. Toto zariadenie dávalo príkaz na stiahnutie (Kód 6.1). Prijaté dáta neboli ukladané na disk ale skartované pomocou presmerovania do pseudozariadenia *devnull*. Štatistiky boli merané každých 0.1s pomocou nástroja *glances* a zobrazené pomocou nástroja *grafana*.

Kód 6.1: Sťiahnutie súboru na klientovi

```
1 curl -k https://192.168.1.100/fedora/output.tar > /dev/null
```

Počas testovania boli sledované hodnoty zachytených paketov a hodnoty stratených paketov, ktoré MitM Sonda nestihla spracovať. Po meraniach bola nameraná priemerná strata 3.6% paketov. Záchyt paketov bol vykonávaný na *Intel* sieťovej karte.

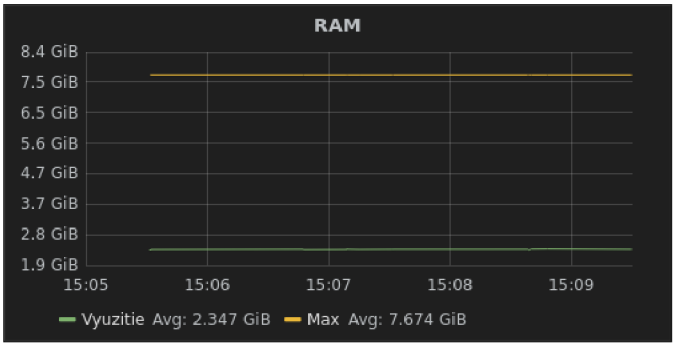
Využitie zdrojov

V ďalšom teste bola meraná rýchlosť príjímania dát, odosielania dát, využitie pamäti RAM a zaťaženie CPU na sonde. Počas normálneho behu zariadenia, bez spusteného útoku bolo namerané priemerné využitie ram 1.86GB. Z grafu 6.8 je možné vidieť, že aplikácia nieje veľmi náročná na pamäť RAM, keďže počas útoku bolo využité priemerne 2.347 GiB. Z grafu využitia CPU počas útoku (Obrázok 6.9), je možné taktiež vidieť, že má sonda taktiež dostatočnú rezervu.

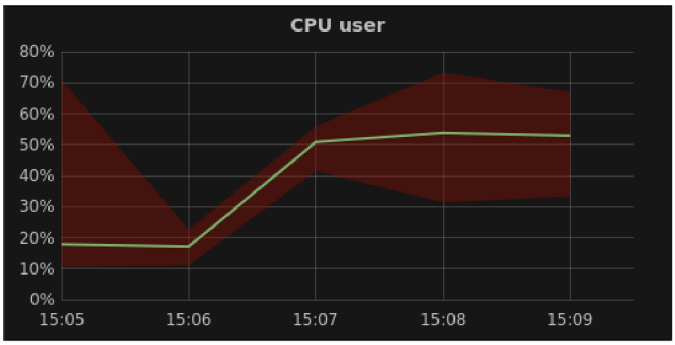
Rýchlosť prenosu dát

Počas experimentu bola najskôr na klientskom zariadení nameraná rýchlosť prenosu vyššie spomínaného súboru *output.tar* medzi klientom a serverom bez zapojenia sondy. Rýchlosť dosahovala priemerne 969Mbps počas sťahovania (Obrázok 6.10).

V ďalšom teste bola zapojená MitM Sonda so spusteným útokom a bol zopakovaný rovnaký scénar so sťahovaním súboru. Nameraná priemerná hodnota (Obrázok 6.11) bola rovnaká ako v predošlom zapojení. Súbor bol medzi klientom a serverom sťahovaný paralelne 3 krát. Po dokončení prenosu bolo na sonde zaznamenaných 30 pcap súborov a ich

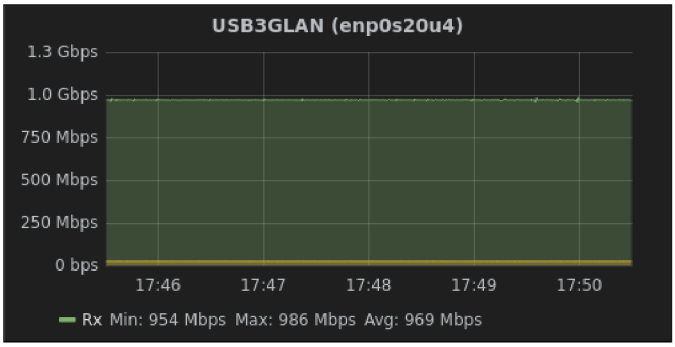


Obr. 6.8: Využitie RAM pamäti počas MitM útoku



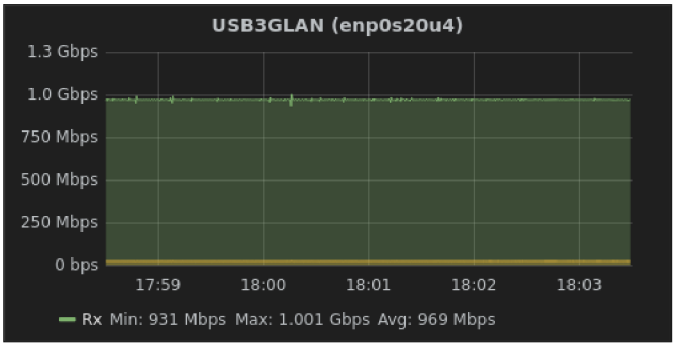
Obr. 6.9: Využitie CPU počas MitM útoku

veľkosť bola 36.7GB. Navyše na sonde bol uložený súbor s hlavnými kľúčmi relácií, ktorý mal veľkosť 4.2Kb.

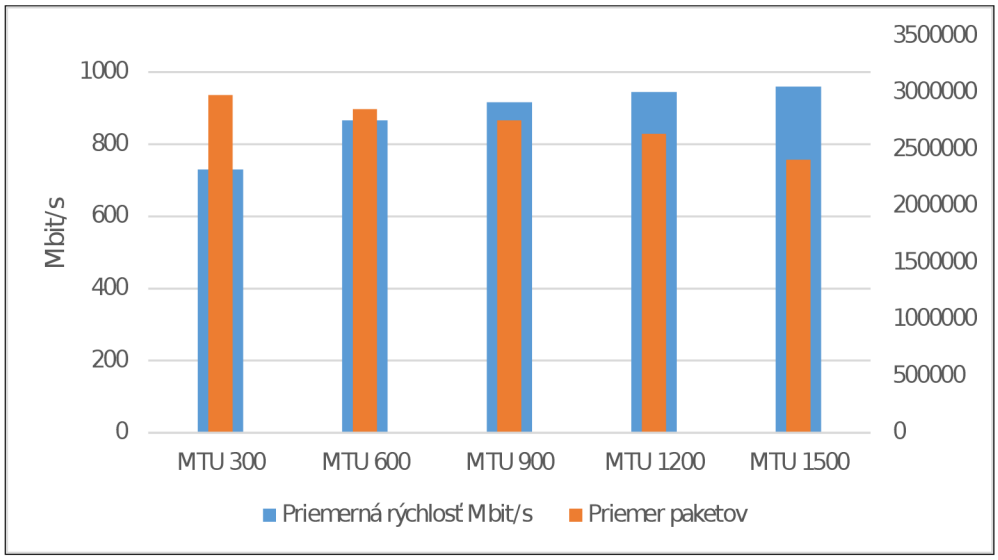


Obr. 6.10: Rýchlosť sťahovania bez zapojenia MitM sondy.

Rýchlosti prenosu boli navyše merané pri rôznych nastaveniach veľkosti paketov, čo spôsobilo rôzny počet prenášaných paketov počas sťahovania súboru. Zmenšenie paketov bolo dosiahnuté pomocou zmeny nastavenia MTU na spomínanom servery. Postupne boli vykonávané merania s veľkosťou MTU 1500, 1200, 900, 600 a 300. Po každej zmene bolo vykonaných 5 meraní. So zníženou veľkosťou paketov sa zvyšoval celkový počet prenesených paketov a priemerná rýchlosť prenosu postupne klesala (Obrázok 6.12). Okrem toho sa znižovali aj maximálna a minimálna rýchlosť (Obrázok 6.13). Podrobnejšie hodnoty po meraní viz. príloha C



Obr. 6.11: Rýchlosť sťahovania počas MitM útoku.



Obr. 6.12: Rýchlosti prenosu pri rôznych veľkostiach paketov

	MTU 300	MTU 600	MTU 900	MTU 1200	MTU 1500
Priemerná rýchlosť bit/s	729 932 642	865 673 623	915 914 557	944 657 049	959 054 866
Maximálna Rýchlosť	858 261 133	972 783 685	964 540 410	984 803 796	993 192 297
Minimálna rýchlosť	25 951 615	17 124 000	68 144	278 613 066	504 083
Medián	743 881 978	885 678 760	929 932 400	959 260 669	969 189 218
Priemer paketov/s	45 124	46 167	46 868	46 507	44 493

Obr. 6.13: Hodnoty zmerané počas zmien MTU

6.1 Zhrnutie

Zo záťažových testov je možné vyvodiť, že na to aby sonda stíhala zaznamenávať vysoko rýchlostné linky, je veľmi dobre vybavená. Sondu by bolo možné minimalizovať, no pre správny beh by boli minimálne požiadavky na pamäť RAM približne 3GB. Po preskúmaní príčiny spomínanej straty paketov, by pri minimalizácii mohlo byť výhodné použitie procesoru s nižším výkonom, ale s vyšším počtom jadier. Pri použití vyššieho počtu jadier by bolo možné na príjímanie paketov použiť iné jadro ako na ukladanie paketov čím by sa zvýšila rýchlosť ich záznamu a nemuseli by čakať v rade na spracovanie. Ďalšie jadro by mohlo byť použité na beh MitM Automatizéru. Týmto rozdelením by sa znížili straty paketov. spracovanie paketov paralelne a taktiež by mohol MitM Automatizér bežať na inom jadre.

Taktiež bolo zistené že pri vysokej prenosovej rýchlosti a pri veľkom vyťažení odpočúvanej sieti by znamenalo veľmi rýchle zaplnenie disku sondy.

Kapitola 7

Záver

V tejto práci sme sa zamerali na preskúmanie kryptografických protokolov SSL a TLS, ktoré sú dnes používané na šifrovanie sieťovej komunikácie na šiestej prezentačnej vrstve OSI modelu. Cieľom práce bolo vytvoriť aplikáciu, ktorá umožní jednoduché a automatické spustenie MitM útoku čo nám umožní prístup k dešifrovanému obsahu webovej komunikácie medzi klientmi a servermi. Tento nástroj je možné použiť na operačnom systéme Linux. Táto aplikácia je nainštalovaná na sonde v laboratóriách Fakulty Informačných Technologií na VUT. K tejto práci bol taktiž vytvorený článok na súťaž Excel@FIT 2016, ktorý je možné si prečítať v prílohe D.

Navyše tu sú popísane rôzne dostupné nástroje určené na odpočúvanie v SSL/TLS šifrovanom pripojení, napr. *SSLsplit*, *mitmproxy*, and *MITMf*. Sú tu taktiež spomenuté rôzne scenáre používane pre dešiforvanie zychetenej SSL/TLS komunikácie pomocou privátneho kľúča alebo pomocou *hlavných hesiel* SSL/TLS relácií. Spomenutá je zvolená hárdverová konfigurácia MitM sondy, ktorá je použitá pre automatizáciu MitM útoku. Ďalej sú popísané možné druhy zapojenia sondy do siete LAN pomocou *arpspoof* alebo do siete medzi klienta a Internet pomocou vytvorenia mostu zo spomínaných rozhraní sondy.

V kapitole 5 je možné sa dočítať o automatizačnom skripte MitM Automatizér a čo všetko je nutné pre spustenie MitM útoku, aké rôzne nástroje boli použité (napr. *SSLsplit*, *tcpdump*, *iptables*, atď.). Navyše je tu popísané ako boli jednotlivé nástroje použité a aké triedy a funkcie spravujú tieto nástroje.

Počas testovania sme zistili, že pri použití vlastných CA certifikátov, ktoré neboli importované do analyzovaných zariadení dostávame rôzne upozornenia na rôznych verziách webových prehliadačov a pri niektorých počas návštevy HSTS hostiteľov nebolo vôbec možné pripojenie na požadovaný web server.

Demonštráciu dešifrácie zachytených paketov pomocou *Wireshark nástroja* a rôzne správanie webových prehliadačov je možné videieť v kapitole 6. Tiež je tu možné vidieť rôzne testy a merania straty paketov a rýchlosti prenosu počas MitM útoku.

V budúcnosti by vytvorená aplikácia mohla byť rozšírená o nástroje podporujúce prekonanie HSTS (Sekcia 2.1.5) takýto nástroj je napr *MITMf*. Ďalšie plánované rozšírenie je podpora MitM útoku na SSH protokol alebo napr. *Javascript keylogger*, ktorý dokáže odpočúvať stlačené klávesy na webovej lokalite a rôzne ďalšie voľne dostupné nástroje.

Literatúra

[1] *MITMf*. [online]. [cit. 2016-05-16]. Dostupné z: https://github.com/byt3bl33d3r/MITMf.

[2] *Secure Shell*. [online]. [cit. 2016-05-16]. Dostupné z: https://en.wikipedia.org/wiki/Secure_Shell.

[3] Bědajánek, O.: Infrastruktura veřejných klíčů. 2008.

[4] Davidoff, S.; Ham, J.: *Network forensics: tracking hackers through cyberspace*. Prentice hall New Delhi, 2012.

[5] Dierks, T.: The transport layer security (TLS) protocol version 1.2. 2008.

[6] Fielding, R.; Gettys, J.; Mogul, J.; aj.: RFC2616: Hypertext Transfer Protocol-HTTP/1.1. W3C. 1999.

[7] Gomzin, S.: *Securing .NET Web Services with SSL: How to Protect "Data in Transit" between Client and Remote Server*. Book-n-share Media, ISBN 9781476064451. URL https://books.google.sk/books?id=QHwiiZBaNc0C

[8] Hartpence, B.: *Packet Guide to Routing and Switching*. Ö'Reilly Media, Inc.", 2011.

[9] Hickman, K.; Elgamal, T.: The SSL protocol. *Netscape Communications Corp*, ročník 501, 1995.

[10] Holz, R.; Sheffer, Y.; Saint-Andre, P.: Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS). 2015.

[11] Jackson, C.; Barth, A.; Hodges, J.: HTTP Strict Transport Security (HSTS). 2012.

[12] Khan, D.: *The Most In-depth Hacker's Guide*. ISBN 9781329727687. URL https://books.google.sk/books?id=sAwiCwAAQBAJ

[13] media, C.: *Latest Mobile Apps and Technology: 6 Top Smartphones to Buy under 10K In 2015*. Series 2, Cloudpeer Media Technologies, 2015, ISBN 9781508658764. URL https://books.google.sk/books?id=rXDVBgAAQBAJ

[14] Oppliger, R.: *SSL and TLS: Theory and Practice*. Artech House, 2014.

[15] Rescorla, E.: Http over tls. 2000.

[16] Ristic, I.: *Bulletproof SSL and TLS: Understanding and Deploying SSL/TLS and PKI to Secure Servers and Web Applications*. Feisty Duck, 2014.

[17] Wagner, D.; Schneier, B.; aj.: Analysis of the SSL 3.0 protocol. In *The Second USENIX Workshop on Electronic Commerce Proceedings*, 1996, s. 29–40.

[18] Zimmermann, H.: OSI Reference Model–The ISO Model of Architecture for Open Systems Interconnection. *Communications, IEEE Transactions on*, Apr 1980: s. 425–432, ISSN 0090-6778, doi:10.1109/TCOM.1980.1094702.

Prílohy

Zoznam príloh

Príloha A

Obsah CD

Na elektronický nosič boli priložené následujúce súčasti bakalárskej práce:

- **src/** – Zdrojové súbory
- **latex/** – Latex zdrojové súbory technickej správy
- **sprava.pdf** – PDF verzia technickej správy
- **README** – Textový dokument obsahujúci inštrukcie ako spustiť nástroj
- **plagat.pdf** – PDF plagátu vytvoreného k Excel@FIT a tejto práci
- **clanok.pdf** – PDF verzia článku z Excel@FIT

Príloha B

Návod na inštaláciu

Pre správne fungovanie MitM Automatizéru je nutné použiť hardvér s dostupnými 2 Ethernetovými vstupmi. Odporúčaný je OS Archlinux pri, ktorom je overené že postup funguje a sú na ňom dostupné všetky potrebné balíčky.

B.1 Inštalácia

Z priloženého CD je treba nakopírovať všetky súbory zo zložky *src*.

Závislosťi

Pred spustením je nutné nainštalovať balíčky, ktoré sú nutné pre správny chod aplikácie a inštalácie:

```
1 pacman -S libevent openssl check bridge-utils \
2     dsniff tcpdump python2-colorlog python2-pyqt4
```

SSLsplit

V zložke *src/sslsplit* je nutné spustiť príkaz *make*, ktorý skompiluje tento nástroj aby ho bolo možné používať.

Spúšťanie

Pred prvým spustením je nutné v zdrojovom súbore *run.py* nutné doplniť správne názvy používaných sieťových kariet. Pri použití s MitM Sondou sú už správne názvy *eno1* a *enp2s0* pred pripravené. Navyše je nutné doplniť správnu cestu do premenných *outtDir* kam sa budú zachytené dáta ukladať. Po správnom doplnení je možné útok spustiť ako *root* užívateľ v zložke *src/* príkazom (pre podrobnejšie informácie o stave je možné použiť *–dbg* parameter):

```
1 ./run.py
```

Pre nápovedu je možné použiť parameter *–help* alebo *-h*, kde sú popísané ďalšie možnosti spustenia.

Príloha C

Tabľuka s výsledkami meraní

	MTU 300	MTU 600	MTU 900	MTU 1200	MTU 1500
Priemerná rýchlosť bit/s	729 932 642	865 673 623	915 914 557	944 657 049	959 054 866
Maximálna Rýchlosť	858 261 133	972 783 685	964 540 410	984 803 796	993 192 297
Minimálna rýchlosť	25 951 615	17 124 000	68 144	278 613 066	504 083
Medián	743 881 978	885 678 760	929 932 400	959 260 669	969 189 218
Smerodatná odchilka	88 179 705	114 055 708	84 995 323	67 258 914	73 798 375
Rozptyl	7 775 660 421 090 090	13 008 704 615 146 100	7 224 204 860 112 600	4 523 761 536 810 110	5 446 200 096 636 810
Priemer paketov/s	45 124	46 167	46 868	46 507	44 493

Obr. C.1: Hodnoty zmerané počas zmien MTU

Príloha D

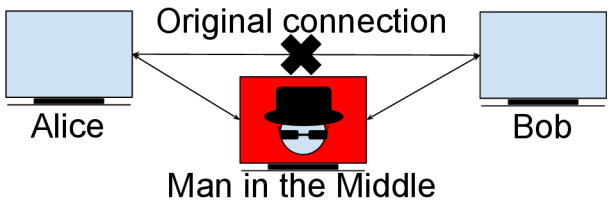
Článok Excel@FIT

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2016
http://excel.fit.vutbr.cz

Automatization of MitM Attack for SSL/TLS Decryption

Marek Marušic*



Abstract
SSL/TLS are protocols used to encrypt network traffic. They provide secure communication between clients and servers. The communication can be intercepted with MitM attack. This paper is aimed to describe the automatization of MitM attack and demonstrate its results. The automatization is done by MitM probe and a python script, which configures the probe and starts the attack. The script is easy to use, without great effort. It takes care of configuration of the probe, then it starts the tools used for network traffic capture and at last it starts MitM tools to do the attack. During the MitM attack, users are warned by client applications about insecure connection. The client applications either provide an option to establish a connection anyway or it forbids clients to establish the connection with insecure parameters. In this paper, the users can learn what are SSL/TLS protocols and about a possibility how to intercept the network traffic encrypted by these protocols.

Keywords: MitM — attack — SSL — TLS — decryption

Supplementary Material: N/A

*xmarus05@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

SSL/TLS encrypted network traffic is nowadays used for secure transmission of various network protocols over an insecure infrastructure of Internet in order to provide secure communication between end points. However, sometimes there is a need to decipher and analyze the communication between criminals or suspects by the government, police or secret services. Existing tools for the decryption are very complicated to use, therefore it would take a great amount of time to perform the decryption by a common user. Furthermore, the user would have to use various tools in order to be able to intercept the communication in networks with various properties (ipv4/ipv6, various cipher suites, various type of connection into the network, etc.). The goal of this work is to provide easy to use solution, which supports various network properties for interception of the SSL/TLS traffic.

A great number of the web hosts on the Internet use SSL/TLS protocols (see Subsection 2.1). It would take a great amount of time to decipher the captured SSL/TLS traffic with brute force attack. This means that we need a better solution with faster and easier access to the decrypted data. There is a lack of a tool witch would support both ipv4 and ipv6, capture the packets into pcap file (most common format for network analysis) with ability to decipher them, support DH key exchange (see Subsection 2.1), support various types of connection into the network, with ability to act as transparent proxy and last but not least with an easy usage for the users.

There are many tools focused on deciphering the

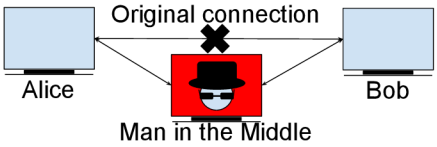


Figure 1. An illustration of the Man in the Middle attack

SSL/TLS traffic, which use various scenarios for the decryption. A great number of the tools are open source and can be used free of charge. Few of these tools will be described in a Subsection 2.4. These tools often require complicated setup of a device and of the tool itself.

During this work there was set up a MitM probe (see Section 3) which is very easy to use. The probe uses a python script which will automatically configure the probe and run the MitM attack. There is used SSL/TLS intercepting proxy (see Section 2), which terminates the connection between clients and servers and establishes its own connections with both of them. Then it forwards the data from clients to servers and vice versa. For clients, the proxy acts as visited server with its own certificate. The probe captures all network packets into its SSD disk and the captured network traffic is then deciphered by a private key of the SSL/TLS proxy or with captured *master secrets* of SSL/TLS sessions.

2. Man in the Middle Attack

The classic Man in the Middle (MitM) attack (see Figure 1) is a scenario where an attacker intrudes into server-client connections. This attack allows the attacker to eavesdrop and modify network traffic and connections while the communicating endpoints believe that they are talking with each other.

The attacker must have access to the network that endpoints are using and all the intercepted traffic has to go through the MitM device. There are few ways how to achieve this:

1. On LAN network this can be done with a *arpspoof* tool, which redirects packets from a target host (or all hosts) on the LAN intended for another host on the LAN (see Subsection 3.2).
2. The MitM device can be connected to the upstream network, e.g., ISP connection (see Subsection 3.1).

2.1 SSL/TLS Protocols

The major enemies of the MitM attack are SSL and TLS protocols. They are ”cryptographic protocols designed to provide secure communication over insecure infrastructure.” [1] ”The primary goal of these protocols is to provide privacy and data integrity between two communicating applications.” [2] This means that the intercepted network traffic packets are useless for the attacker hence the data are encrypted and only client and server can access the data.

Client		Server
Client hello	$\longrightarrow$	
	$\longleftarrow$	Server hello
	$\longleftarrow$	Server certificate*
	$\longleftarrow$	Server key exchange*
	$\longleftarrow$	Certificate request*
	$\longleftarrow$	Server hello done
Client certificate*	$\longrightarrow$	
Client key exchange	$\longrightarrow$	
Certificate verify*	$\longrightarrow$	
[Change cipher spec]	$\longrightarrow$	
Finished	$\longrightarrow$	
	$\longleftarrow$	[Change cipher spec]
	$\longleftarrow$	Finished
Application Data	$\longleftrightarrow$	Application Data

Table 1. SSL/TLS Handshake [2] (*optional message, [Change cipher spec] is an independent protocol)

”When a SSL/TLS client and a server first start communicating, they agree on cryptographic parameters of the session produced by the SSL/TLS Handshake Protocol” (illustrated in Table 1). ”They use public-key encryption techniques to generate shared secrets.” [2] There are different algorithms for the exchange of generated shared secrets over network e.g RSA [1] (page 36) , Diffie–Hellman (DH)[3], ephemeral Diffie-Hellman (DHE), Elliptic Curve Diffie-Hellman (ECDH), ephemeral Elliptic Curve Diffie-Hellman (ECDHE) etc. Modern web applications prefer the most secure ones, i.e., DHE and ECDHE key exchange. The public-key encryption techniques use asymmetric encryption. The shared secrets are then used in symmetric encryption of the network traffic. The public key is a part of a host certificate. The certificate is a digital document, which bears the public key, a digital signature of the certificate issuer (CA authority) and the information about its validity and owner.

2.2 HTTP Strict Transport Security (HSTS)

Another enemy of the MitM attacks is the HSTS protocol, which enforces usage of the SSL/TLS encryption between the client and server and forbids the client to accept an untrusted or a fake certificate as well. The client software will warn the user about possible MitM attack and block all the communication with HSTS host until there is a secure connection. The HSTS can be enabled through a *Strict-Transport-Security* HTTP header or through preloaded list of HSTS hosts in the

client's software application, e.g., web browser. [4]

2.3 SSL/TLS Decryption

There are two common scenarios how to get to decrypted data of the captured SSL/TLS network traffic. In the first case, we need to capture whole SSL/TLS Handshake and have the server's private key, which was used during the Handshake. This way we can obtain the shared secrets, which was used for the symmetric encryption of the session. However, this scenario does not work with key exchange algorithms based on a Diffie-Hellman algorithm, where the shared secrets are not transmitted through the network, but they are calculated on the endpoint devices.

The second option is to capture connections with SSL/TLS proxy. The proxy will split the connection of the intercepted client and server to downstream and upstream connections. Downstream communication between client and proxy will use proxy's certificate. The upstream connection between proxy and server will use server's certificate. This solution eliminates the problem with DH key exchange since the sessions were negotiated with the proxy, thus it has access to all the parameters of the sessions. [5]

2.4 MitM Tools for SSL/TLS Decryption

Besides the classic MitM attack, we need to decrypt intercepted SSL/TLS network traffic. For this purpose, we can use tools implementing the aforementioned decryption scenarios. This can be called sslplitting. There is another way how to get to the decrypted data of the SSL/TLS network traffic and it is called sslstripping. Sslstripping also works as SSL/TLS proxy, however only the communication between proxy and server is encrypted and the communication between client and proxy is not encrypted. In the following, I will discuss few open-source tools in this section.

SSLsplit[1] works as a SSL/TLS proxy between client and server. "*SSLsplit* supports plain TCP, plain SSL, HTTP and HTTPS connections over both IPv4 and IPv6. For SSL and HTTPS connections, *SSLsplit* generates and signs forged X509v3 certificates on-the-fly, based on the original server certificate subject DN and subjectAltName extension." [6] With this tool we can not intercept HSTS preloaded hosts since client applications disallow clients to accept untrusted certificates.

Mitmproxy[2] is "an interactive, SSL-capable man-in-the-middle proxy for HTTP with a console interface." [7] This tool captures the data from the application layer as well as *SSLsplit*. However, it uses its own certificate with the public and private key for the SSL/TLS communication, which can be used for decryption of the captured packets from tcpdump.

MITMf[3] **and Bettercap**[4] use *SSLstrip+*[5] and special DNS server *dns2proxy*[6] to implement partial HSTS bypass. The main difference is the scenario used in SSL/TLS proxy of this tool. The SSL/TLS connections are also terminated by them, however, the downstream connection between client and attacker does not use SSL/TLS encryption and remains decrypted. The partial HSTS bypass redirects the client from domain name of the visited web host to a fake domain name by sending HTTP redirection request. The client is then redirected to a domain name with extra *w* in *www* or *web.* in the domain name e.g. *web.example.com*. This way the web host is not considered as a member of HSTS preloaded hosts list and the client can access the web host without SSL/TLS. The fake domain names are then resolved to real and correct IP addresses by the special DNS server, which expects these changes in the domain names. The downside of this attack is that the client has to start the connection over HTTP due to the need of HTTP redirection.

The main disadvantage of the *SSLsplit* and *mitmproxy* is a poor support of HSTS bypass for HSTS preloaded hosts, compared to *MITMf* and *bettercap* which support HSTS bypass and allow clients to connect to HSTS preloaded hosts.

All mentioned tools are not able to save the network traffic into a pcap file format since they work with the application layer of the *ISO/OSI* network model [8]. However, we can use a tcpdump to capture the data into a pcap format and decrypt captured packets using the private key of the used fake certificate. This is not possible with cipher suites using DH key exchange algorithm. However, *mitmproxy* can save the *master secrets* of sessions into a file with NSS Key Log Format[7] and use this file for decryption of the data instead of the private key, which eliminates the problem with DH key exchange.

[1] https://www.roe.ch/SSLsplit
[2] https://mitmproxy.org/
[3] https://github.com/byt3bl33d3r/MITMf
[4] https://www.bettercap.org/
[5] https://github.com/byt3bl33d3r/sslstrip2
[6] https://github.com/LeonardoNve/dns2proxy
[7] https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS/Key_Log_Format

	SSLsplit	Mitmproxy	MITMf	bettercap	MitM Probe
ipv4	✔	✔	✔	✔	✔
ipv6	✔	✘	✔	✔	✔
master keys logging	✘	✔	✘	✘	✔
partial HSTS bypass	✘	✘	✔	✔	✘*
sslsplitting	✔	✔	✘	✘	✔
sslstripping	✘	✘	✔	✔	✘*
arpspoof	✘	✘	✔	✔	✔
bridge	✘	✘	✘	✘	✔
CA certificates	✘	✔	✘	✘	✔
pcap output	✘	✘	✘	✔	✔
auto traffic redirection	✘	✘	✔	✔	✔

Figure 2. Comparison of the MitM tools (*these properties should be also implemented in the future)

3. SSL/TLS Intercepting Probe

The purpose of the probe is to simplify and automatize the MitM attack on the connected network. The goal is to be able to connect input and output Ethernet cables, login to OS on the probe and start the MitM attack just by a double click on an icon. In the future, the used software should provide a GUI for greater scalability with a simple control. The setup and start of the MitM are provided by my python script, which starts and configures all the necessary software and settings.

The OS is installed on a 32GB USB 3.0 flash for an easy transportation. The probe contains SSD 120GB disk, 3.2 GHz Intel Pentium G3258 processor and 8GB RAM for fast packet capture. The greatest criterion during an HW selection process was the motherboard. It should own two GLAN interfaces with very good throughput in order to be able intercept and capture communications of the whole subnet. The selected one is Mini ITX Intel Motherboard GIGABYTE GA-Z97N-WIFI - Intel Z97 with Intel (*eno1*) and Realtek (*enp2s0*) GLAN interfaces. Moreover, it supports Wifi connection, therefore, we are able to perform the MitM attack wirelessly as well.

I have used Archlinux[8] as OS due to its powerful user repositories, which provide most of the used tools and packages, without the need to compile them from sources manually. I have subsequently installed all the necessary packages needed to realize the MitM attack. I will describe all the required packages and their usage in this document. The python script, responsible for the MitM attack initiation, encapsulates setup of the settings and tools which will be described in this Section.

With OpenSSL tool, I have created CA certificate with public and private key pair, which are required by *SSLsplit*. This CA certificate is then used to sign on-fly created certificates for the visited hosts. Users are able to use their own CA certificate.

[8] https://www.archlinux.org/

3.1 Bridge

This setup is needed in case of access to the upstream Ethernet cable before the intercepted LAN or Host (see Figure 3). The scenario is considered as default setup of the MitM probe. Bridge is created with Linux tool *brctl*, which will create a bridge between the 2 GLAN interfaces of the probe. It is necessary to connect an Ethernet cable with the Internet connection to eno1 interface and to the second interface enp2s0 the Ethernet cable with the connection to the intercepted network. The bridge acts as a virtual network switch working transparently, this behavior can be called transparent firewall. This way the probe will forward all the traffic from the client to the Internet and vice versa. With this setup, we have access to all packets transmitted from the clients and servers without the need to use any additional tools (e.g *arpspoof*) to redirect the traffic into our interfaces. For the correct behavior it is necessary to start *dhclient* process as well, which will enable clients to automatically acquire IP addresses.

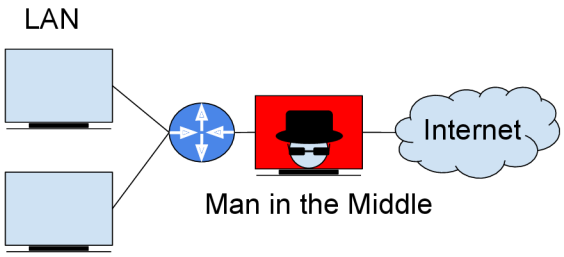


Figure 3. An illustration of the MitM on LAN

3.2 Arpspoof

Arpspoof is required to use, in case we have no access to the upstream network, instead, we have access to LAN connection (see Figure 4). The client and the MitM are connected to the default gateway. The default gateway is a network device which connects LAN with the Internet. It forwards the traffic from the local subnet to the other connected subnets. The tool is spoofing Address Resolution Protocol (ARP) messages to fake the ARP records in ARP tables of devices in the network. ARP is used for resolution of IP address to a physical Ethernet (MAC) address [9]. The IP address of the gateway is then mapped to MAC address of attacker's interface rather than the original MAC address of the gateway. It allows users to choose whether to spoof the whole LAN or a specific IP address as well. It is not recommended to attempt to spoof busy LAN networks with an insufficient (slow) network card, due to possible denial of service on the network or rapid

slowdown of the network. This approach is unfortunately not as fast, reliable and transparent as the bridge solution.

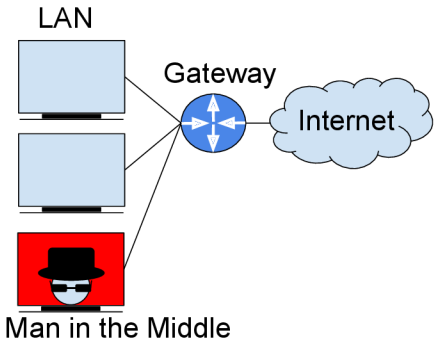


Figure 4. An illustration of the MitM on LAN

3.3 Iptables and Ip6tables

They are "administration tools for IPv4/IPv6 packet filtering and NAT" and "are used to set up, maintain, and inspect the tables of IPv4 and IPv6 packet filter rules in the Linux kernel." [10]. It is able to redirect the incoming and outgoing traffic to the specified port. I have used this tool to create redirection of all HTTPS (tcp port 443) and HTTP (tcp port 80) network traffic to the address and port where the *SSLsplit* is listening and waiting for incoming traffic. Besides the use of *iptables*, users have to enable a packet forwarding on the used interface in order to enable a communication between clients and servers.

3.4 Tcpdump

This tool is used to capture all the traffic of the intercepted network. It uses pcap file format readable by a packet capture library libpcap[9], which is used in network traffic analyzer tools, e.g., Wireshark. All captured data will be stored on the SSD disk. File size is limited to 100MB per file. Users can set a limit for a number of stored files and after reaching max file count, it will start rewriting the created files from the oldest file. There can be set a filter on packets as well to eliminate unnecessary packets and capture just relevant ones, e.g., TCP and TLS packets. This is useful in case of low disk space or for a longer monitoring.

3.5 SSLsplit

For the MitM attack I have chosen the *SSLsplit* tool, which is written in C programming language, thus it is very fast. It supports all TLS and TCP traffic, compared to other mentioned tools, which provide only HTTPS and HTTP support. The traffic redirection is done by linux firewall called *iptables* and *ip6tables* for ipv6 to *SSLsplit*'s address and port. For HTTPS traffic over ipv4, it is address 0.0.0.0 and port 10443, for traffic over ipv6, it is listening on address ::1 and port 10443. The same addresses are used for HTTP traffic, however, there is a difference in used port number 10080. As an NAT engine is used *tproxy*, due to its support of ipv4 and ipv6. It is the only engine with ipv6 support on Linux OS, however on FreeBSD, OpenBSD there are few more, e.g., *pf* or *ipfw*. Aforementioned generated CA keys, i.e., ca.crt, ca.key, ca.pem, are used as root authority keys in *SSLsplit*. It is necessary to use private key with parameter -K (should be *.key or *.pem key format) in order to be able to decrypt captured packets. However, this is only possible when cipher suites without DH key exchange algorithm are used, nonetheless, the modern client applications prefer DH key exchange cipher suites. The script provides two solutions to this problem. The first solution forbids DH cipher-suties and offers only ciphersuites using RSA key exchange algorithm to the client application. Another solution is to log *master secrets* of sessions and use them for decryption of the captured packets. I had to implement this feature into *SSLsplit*, and now it can log the *master secrets* with parameter -A into the file. This is the second solution providing the ability to decrypt all the captured data regardless the used key exchange algorithm.

4. Experiments

The basic experiment is to connect MitM probe between the upstream and downstream network (see Subsection 3.1) and run the *SSLsplit* by the aforementioned python script. Clients will get warning about untrusted certificate used by the visited host in the address bar of the web browser (see Figure 5) and there will be a more explanatory warning instead of a visited web page content. This warning will provide the user with

Figure 5. Insecure HTTPS connection

advanced options. There is an option during the visit of the ordinary host without HSTS to accept the untrusted certificate and proceed to the web page with the security risks (see Figure 6) example of web host without HSTS is a *www.seznam.cz*. This warning will be present every time the client visits an unvisited webpage, during the MitM attack. In case, the host is preloaded in the HSTS list of the web browser there is

[9] http://www.tcpdump.org/manpages/pcap.3pcap.html

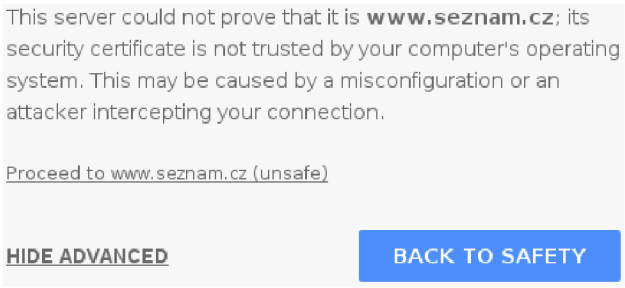


Figure 6. Warning with option to continue with insecure connection

not any option to accept and proceed to the web page. This behaviour can be observed during an attempt to visit *google.com* under MitM attack (see Figure 7). The warning about untrusted certificate disappears at

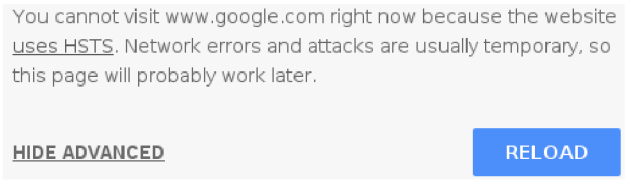


Figure 7. Warning without option to continue to HSTS web page over insecure connection

the moment the client imports the CA authority inside of his OS or client application. There is the possibility to use HSTS bypass from the aforementioned *bettercap* as well. However, the connection is just over the plain unecrypted HTTP (see Figure 8). Captured pcap

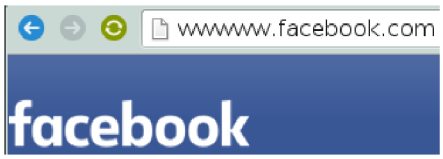


Figure 8. HSTS bypass on *wwwww.facebook.com*

files can be opened and decrypted with commonly used tool Wireshark along with the created private key or with the captured SSL *master secrets* log file. The *master secrets* log file or RSA keys can be applied in the Preferences - Protocols - SSL menu. Decrypted data of the SSL/TLS packets can be examined with the Wireshark. Users have to select the Decrypted SSL data option in the selected packet (see Figure 9). The decrypted HTTP data are very usefull in the further analysis, where we can discover what are the subjects chatting about, doing on the Internet and intercept user credentials and sessions. The application layer data can be examined with the option *Follow SSL stream* (see Figure 10), where it is possible to see the decrypted raw HTTP protocol data.

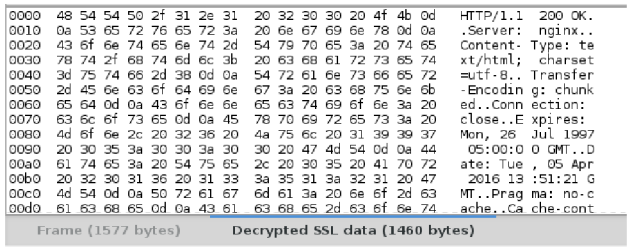


Figure 9. Decrypted SSL packet

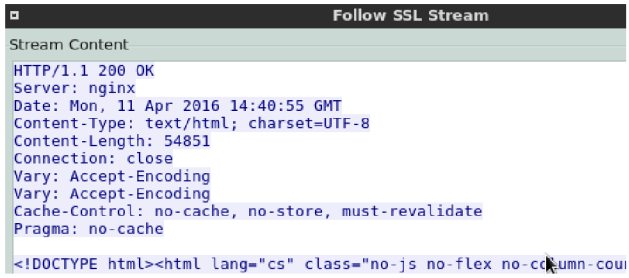


Figure 10. Decrypted HTTP response in Wireshark

5. Contributions

I Made a research about abilities of the MitM tools and created comparison in the survey (see Figure 2). Choose the best tool for the automatization which support the most of the properties. Implemented the master secrets logging into *SSLsplit*, in order to solve the problem with DH key exchange. Created and preinstalled a script into MitM probe with a support for most of the aforementioned properties. The script introduces an ability to work as a transparent proxy and ability to automatically capture the data into pcap file format, which is the most used format in network analysis. Last but not least the script and MitM probe is very easy to use.

6. Conclusions

The paper describes protocols SSL/TLS and MitM attack aimed to intercept SSL/TLS encrypted traffic. Scenarios used to decrypt the SSL/TLS traffic with private key or with the *master secrets* log file of SSL/TLS sessions. Some of MitM attacks and tools, e.g., *SSLsplit*, *mitmproxy*, and *MITMf* capable of performing the MitM attack. Hardware and software setup of MitM probe, which is used to automatize SSL/TLS interception. There was a demonstration how to connect the probe to the LAN by usage of arpspoof or to the upstream network by usage of the bridge.

There was an illustration how to get to deciphered packets of the SSL/TLS traffic. Variations of the MitM attack and their results, e.g, web browser warnings (see Figure 6) and restrictions (see Figure 7) were shown. The paper illustrates what HSTS is and how HSTS can be partially defeated by *MITMf* and *bettercap* as well.

In order to set up a MitM probe I had to research

and learn how to use all mentioned tools used in the script. I have selected HW and OS. Installed the OS with necessary tools. Created script for automatic start of the attack. The script deals with DH key exchange by forcing the RSA key exchange. I have implemented *master secrets* logging into the *SSLsplit* as well.

In the future, the python script for automatization should support HSTS bypass (see Subsection 2.2) and JavaScript keylogger as well. These functionalities should be implemented by adding corresponding modules from *MITMf* and from other open-source tools.

Acknowledgements

I would like to thank my supervisor Ing. Jan Pluskal for his help.

References

[1] Ivan Ristic. *Bulletproof SSL and TLS: Understanding and Deploying SSL/TLS and PKI to Secure Servers and Web Applications*. Feisty Duck, 2014.

[2] Tim Dierks. The transport layer security (tls) protocol version 1.2. 2008.

[3] Eric Rescorla. Diffie-hellman key agreement method. 1999.

[4] Collin Jackson, Adam Barth, and Jeff Hodges. Http strict transport security (hsts). 2012.

[5] Sherri Davidoff and Jonathan Ham. *Network forensics: tracking hackers through cyberspace*. Prentice hall New Delhi, 2012.

[6] Daniel Roethlisberger. Sslsplit - transparent ssl/tls interception. https://www.roe.ch/SSLsplit, April 2016.

[7] Aldo Cortesi. mitmproxy. https://mitmproxy.org/.

[8] H. Zimmermann. Osi reference model–the iso model of architecture for open systems interconnection. *Communications, IEEE Transactions on*, 1980.

[9] Bruce Hartpence. *Packet guide to core network protocols*. O'Reilly Media, Inc., 2011.

[10] Herve Eychenne. iptables man page, 2002.