Excel@FIT 2015
http://excel.fit.vutbr.cz

Získanie prístupu k šifrovanej komunikácii

Miroslav Slivka*

Abstrakt
Táto práca sa zaoberá sprístupnením šifrovanej komunikácie pre nástroj Netfox, ktorý vzniká na Fakulte informačních technologií, VUT v Brně pod záštitou bezpečnostného výskumu (VG20102015022) - SEC6NET. Okrem analýzy SSL/TLS protokolu je v práci popísaný spôsob útoku na šifrované spojenie, návrh a implementácia modulu, ktorý za pomoci súkromného kľúča serveru dešifruje danú SSL/TLS reláciu a testovanie implementovaného riešenia. Algoritmy šifrovania poskytla nástroju externá knižnica Bouncy Castle. Výsledný modul sprístupňuje šifrovanú komunikáciu pre ďalšie spracovanie a extrakciu dát v dešifrovanej podobe.

Kľúčové slová: SSL — TLS — šifrovaná komunikácia — Netfox

Priložené materiály: *N/A*

*xslivk02@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Úvod

Množstvo komunikácie na dnešnom Internete je prenášané šifrovane. K zavedeniu šifrovania viedli napríklad regulačné opatrenia, ktoré vyžadujú aby citlivé informáie boli prenášané šifrovane. Taktiež šifrované spojenie potrebovali ľudia, ktorí musia pristupovať na internet anonymne, napríklad obyvatelia Číny. Dnes asi najpoužívanejším protokolom, ktorý poskytuje šifrované spojenie, je protokol TLS. Okrem šifrovanej komunikácie tento protokol zabezpečí, aby sa komunikujúce strany dohodli na spoločnom tajnom kľúči symetrickej kryptografie pomocou asymetrickej kryptografie, prípadne sa navzájom autentizovali. Asymetrická kryptografia vďaka svojím vlastnostiam slúži aj na autentizovanie.

Tento projekt vzniká ako TLS/SSL dešifrovací modul väčšieho nástroja Netfox.Framework[1]. Nástroje rodiny Netfox vznikajú za účelom analýzy zachytenej komunikácie. Na rozdiel napríklad od nástroju Wireshark, ktorý sa používa na analýzu prevádzky na sieťovom rozhraní tak, ako chodí na sieti, nástroje tohto projektu sa snažia zo zachytenej komunikácie extrahovať aplikačné dáta a tie analyzovať.

Hľadať nedostatky v samotných algoritmoch alebo použiť útok hrubou silou by zabralo množstvo času. Pre spracovanie v reálnom čase je nutné hľadať iné možnosti. Ako efektívne riešenie sa ukázalo získať prístup k súkromnému kľúču serveru a pomocou neho potom danú TLS/SSL reláciu dešifrovať. Táto práca rieši problém ako sa dostať k súkromnému kľúču serveru, ku ktorému normálne nemáme prístup a potom tento kľúč použiť na získanie ostatných parametrov relácie a túto reláciu pomocou získaných znalostí dešifrovať.

Doposiaľ projekt Netfox dešifrovanie neriešil. S využitím tohto modulu budú nástroje rodiny Netfox mať možnosť analyzovať aj šifrované dáta. Inšpiráciu pri implementovaní je možné do určitej miery nabrať v projekte Wireshark, ktorého zdrojové kódy sú verejne dostupné. Avšak implementačne sú tieto dva projekty pomerne odlišné.

Prvá kapitola sa zaoberá analýzou SSL/TLS komu-

nikácie. Čitateľa oboznámi so správami vyžadovanými pre zostavenie šifrovaného spojenia a stručne rozoberie jeho princíp. Druhá kapitola pojednáva o útokoch na SSL/TLS komunikáciu. V tretej kapitole je popis algoritmu dešifrovania SSL/TLS relácie ako je implementovaný v module, o ktorom je táto práca. Štvrtá kapitola popisuje testovanie implementovaného modulu. Na prípravu testovacích dát sa vhodne použijú informácie z kapitoly 3.

2. Analýza SSL/TLS komunikácie

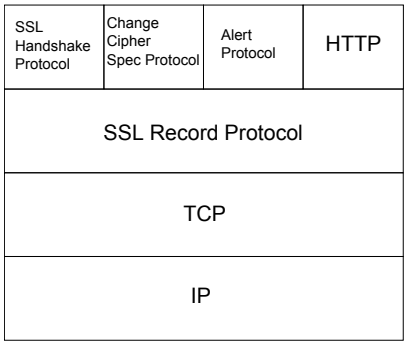
Transport Layer Security (TLS) protokol je IETF štandard navrhnutý na poskytnutie bezpečnej komunikácie cez Internet (najnovšie v RFC 5246 [2]). Predchodcom TLS je Secure Socket Layer (SSL) protokol (viď RFC 6101 [3]). TLS bol navrhnutý ako vylepšnie SSL a stal sa široko používaným protokolom poskytujúcim bezpečnú komunikáciu pre webové aplikácie.

SSL/TLS je vrstvový protokol, ktorý definuje ďalšie protokoly (viď Obrázok 1).

Na vyjednanie parametrov šifrovanej relácie sa používa Handshake protokol. Handshake protokol môžeme rozdeliť na tri časti (viď Obrázok 2): inicializačnú, autentizačnú a ukončenie handshaku.

V inicializačnej fáze klient posiela správu ***Client Hello***, na ktorú server musí odpovedať správou ***Server Hello***. Tieto správy sú použité na zvolenie najbezpečnejších možností, ktoré podporujú obe strany. Vyjednajú sa nasledujúce parametre:

- verzia protokolu,
- ID relácie,
- `CipherSuite`
  - metóda na výmenu kľúčov,
  - autentizačná metóda,
  - symetrická šifra,
  - message authentication code (MAC), na kontrolu integrity,
- kompresný algoritmus,
- náhodné hodnoty (`client.random` a `server.random`)

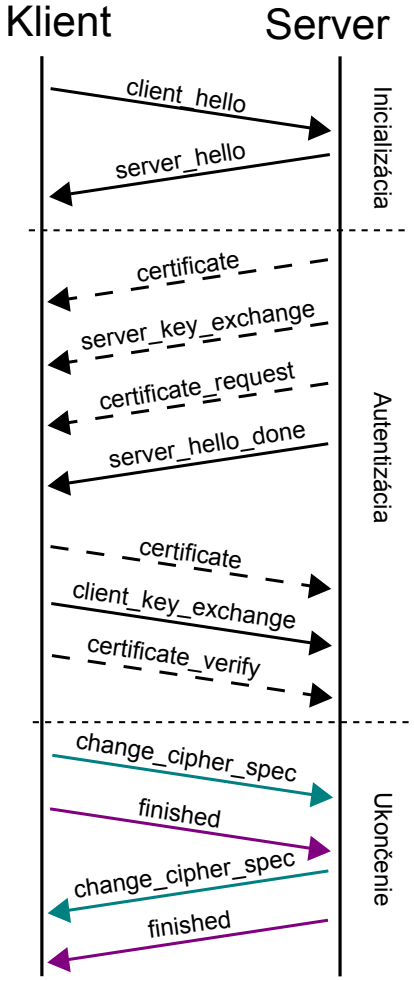


Obrázok 1. Protokolový zásobník SSL. Zdroj: W. Stallings, 1998. [4]

Po výmene *Hello* správ nasleduje autentizačná časť. Táto fáza je v RFC definovaná ako nepovinná. Autentizačnú časť môžeme rozdeliť na autentizáciu serveru a autentizáciu klienta. Autentizácia serveru je obalená medzi správy *Server Hello* a *Server Hello Done*. Autentizácia klienta prebieha vždy až po autentizácii serveru.

Server teda pokračuje v posielaní správy ***Certificate***. Táto správa obsahuje sekvenciu certifikátov. Na začiatku sekvencie je odosielateľov certifikát a každý ďalší certifikát musí autentizovať ten predchádzajúci. Keďže validácia certifikátov je založená na koreňových certifikátoch, ktoré nie sú podpísané, tak koreňový certifikát v tejto sekvencii nie je. Očakáva sa, že druhá strana tento certifikát má.

Následne môže byť odoslaná správa ***Server Key Exchange***, ak server nemá certifikát alebo certifikát je určený len pre podpisovanie. Táto správa teda obsahuje nejaký verejný kľúč, ku ktorému má server súkromný kľúč.



Obrázok 2. Prehľad komunikácie SSL/TLS handshake

Server môže vyžiadať autentizáciu po klientovi

pomocou správy ***Certificate Request***. Následne server odošle správu ***Server Hello Done***, ktorá indikuje ukončenie inicializačnej fázy handshake protokolu.

Na správu *Certificate Request* klient musí odpovedať správou ***Certificate***. Inak je odoslaná správa ***Client Key Exchange***, ktorej obsahom je zašifrovaná hodnota pre-master secret verejným kľúčom algoritmu zvoleného v inicializačnej fáze. V prípade použitia Diffie-Hellman Ephemeral [5] správa obsahuje zašifrovaný verejný exponent tohto algoritmu. Pri použití statickej verzie Diffie-Hellman, klient posiela certifikát obsahujúci statický exponent a táto správa musí byť prázdna. Správa ***Certificate Verify*** sa posiela ak klient poslal certifikát s možnosťou podpisu. Touto správou klient potvrdí, že je držiteľom súkromného kľúča toho certifikátu. V tejto správe sa použijú všetky správy handshaku odoslané a prijaté do tejto chvíle.

Pomocou hodnoty *pre-master secret* sa vygeneruje *master secret* používaný na vypočítanie tzv. *key material*. Z *key material* sa potom časť použije ako kľúč symetrickej šifry, časť ako inicializačný vektor a časť ako kľúč HMAC funkcie.

V ukončovacej fáze sa odosiela správa ***Change Cipher Spec***, ktorá oznamuje, že nasledujúce správy budú odosielané šifrovane. Táto správa nepatrí do handshake protokolu.

Správa ***Finished*** je poslednou správou handshake protokolu a zároveň prvou šifrovanou správou. Je šifrovaná šifrou, ktorá je inicializovaná dohodnutými parametrami. Ako odpoveď server odošle tiež správu *Change Cipher Spec* a rovnako aj *Finished* šifrovanú vyjednanými parametrami. Po tejto výmene klient a server majú nadviazavané šifrované spojenie a môžu pomocou neho komunikovať.

Po ustanovení relácie sa šifrované dáta prenášajú *Record* protokolom.

3. Útok na SSL/TLS komunikáciu

Pre získanie prístupu k dátam prenášaným v TLS/SSL existujú dva základné prístupy, ktoré spomínajú aj S. Davidoff a J. Ham, 2012 [6].

- Prvou možnosťou je získať privátny kľúč serveru na získanie kľúčov relácie a dešifrovanie obsahu. To ale závisí na algoritme použitom na výmenu kľúčov a taktiež je nutný prístup k súkromnému kľúču serveru.
- Druhou možnosťou je ukončiť TLS/SSL reláciu niekde na proxy a nadviazať novú SSL/TLS reláciu smerom od proxy ku klientovy. Pri tomto prístupe je dôležité mať prístup ku klientskej stanici za účelom nainštalovania certifikátu proxy serveru. Inak užívateľ klientskej stanice môže dostať upozornenie od systému na potenciálny útok Man-In-The-Middle.

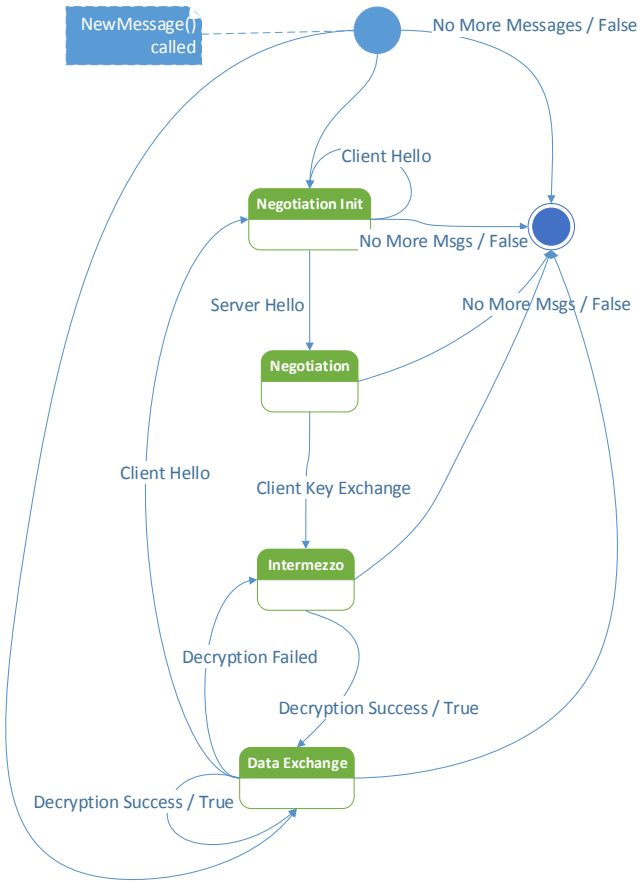
Ak existuje prístup k súkromnému kľúču serveru, pomocou neho je možné dešifrovať hodnotu pre-master secret odoslanú klientom a získať kľúče relácie. Na to je potrebné mať prístup ku kompletnému obsahu handshake fáze TLS/SSL a následnej výmene dát vytvorenou reláciou. Súkromný kľúč serveru pri dešifrovaní prevádzky nepomôže ak bol na výmenu kľúčov zvolený algoritmus Diffie-Hellman Ephemeral. Ten používa špeciálnu schému generovania hodnoty pre-master secret, kde pre každú novú reláciu generuje nové súkromné hodnoty. Táto verzia sa používa v kombinácii s iným asymetrickým algoritmom, ktorý zabezpečí autentizáciu. Teda verejné hodnoty Diffie-Hellman sú podpísané súkromným kľúčom.

Princíp s proxy funguje tak, že celá komunikácia s vonkajším svetom smeruje cez proxy. Ak chce server komunikovať cez TLS/SSL s nejakým klientom, tak TLS/SSL odpovede serveru sú ukončené na proxy a proxy udržuje šifrované spojenie so serverom. Na strane ku klientovi proxy poskytuje falošný certifikát serveru a zostaví druhý TLS/SSL tunel. Potom proxy môže prehliadať prevádzku v dešifrovanej podobe alebo posielať túto prevádzku na iný systém k ďalšej analýze. TLS/SSL je navrhnuté na ochranu proti tomuto typu útoku. Ide vlastne o Man-In-The-Middle. Teda certifikát proxy serveru nebýva podpísaný dôveryhodnou CA a používateľovi na klientskej stanici sa zobrazujú varovania. Tomu sa dá predísť nainštalovaním certifikátu proxy servera medzi dôveryhodné koreňové certifikáty na klientovi (obeti).

SSLsplit[7] je aplikácia vytvorená Daniel Roethlisbergerom a distribuovaná pod zjednodušenou BSD Licenciou. SSLsplit ukončuje TLS/SSL spojenia a vytvára nové smerom k pôvodnej cieľovej adrese a zároveň loguje celú prevádzku. Je možné si zvoliť šifry, ktoré sa majú používať, rovnako aj privátny kľúč, ktorým sa majú podpisovať falšované certifikáty. Na správne fungovanie je potrebné nastaviť IP forwarding[1].

SSLstrip[8] je nástroj vytvorený Moxie Marlinspike a je distribuovaný pod licenciou GNU GPLv3. Implementuje útok podobný ako SSLsplit, s tým rozdielom, že relácia medzi obeťou a útočníkom prebieha nešifrovane.

[1] https://www.kernel.org/doc/Documentation/networking/ip-sysctl.txt



Obrázok 3. Stavový automat modulu PDUDecrypter

4. Návrh a implementácia dešifrovania

Projekt Netfox.Framework[1] je modulárny systém, ktorý sa stará o spracovanie PCAP súborov a ich analýzu. Súčasťou frameworku je reassembling packetov (spojenie súvisiacich rámcov do jednej štruktúry, postaranie sa o znovuzasielanie a pod.) a vytváranie konverzácií. Konverzácia v Netfox.Framework je identifikovaná zdrojovou a cieľovou IP adresou, zdrojovým a cieľovým portom a protokolom sieťovej vrstvy. Konverzácie sú ale iba aproximácie TCP relácií, získaných z komunikácie, ktorá môže byť aj nekompletná. Pred vstupom do modulov, ktoré analyzujú danú komunikáciu sa reassemblované PDU načítajú pomocou modifikovanej triedy `System.IO.Stream`. Trieda implementujúca dešifrovanie bude postavená na rovnaké miesto v spracovaní. Vstupom teda sú reassemblované L7 PDU, ktoré sú obsahom konverzácie. A výstupom sú dešifrované dáta ako prúd bytov.

Na Obrázku 3 je stavový diagram, ktorý reprezentuje princíp algoritmu. Vstupným bodom je metóda `NewMessage()`, ktorá pri každom zavolaní dešifruje jednu správu a pripraví ju na čítanie. Návratová hodnota metódy je `True` ak existuje ďalšia dešifrovaná správa na čítanie, `False` ak nebolo možné dešifrovať žiadnu ďalšiu správu. Táto metóda principiálne implementuje konečný automat, aktuálne so štyrmi stavmi:

- `Init` - decrypter zatiaľ nie je nainicializovaný parametrami relácie,
- `Negotiation` - decrypter je nainicializovaný algoritmami, ale čaká sa na vyjednanie tajného kľúča (správa *Client Key Exchange*),
- `Intermezzo` - medzistav po inicializácii, v ktorom sa dešifruje prvá správa,
- `Data Exchange` - v tomto stave sa dešifrujú správy.

Ak bude potrebné zohľadniť autentizačnú časť, je možné pridať ďalší stav pred `Intermezzo`, v ktorom sa budú spracovávať požadované správy. V momentálnej implementácii je podporovaný algoritmus na výmenu kľúčov RSA, pri ktorom z hľadiska dešifrovania autentizačnú časť nie je potrebné riešiť.

Tento stavový automat je uzavretý v cykle, ktorý sa ukončí po prechode do stavu `Data Exchange`, aby pri každom zavolaní metódy `NewMessage()` bola dešifrovaná nejaká správa.

V prvom stave `Init` sa čaká na *Hello* správy od klienta a od serveru, z ktorých sa získajú potrebné parametre (náhodné čísla, a *CipherSuite*). Z popisu *CipherSuite* sa nainicializuje trieda `KeyDecrypter`, starajúca sa o dešifrovanie kľúča (*pre-master secret*) a trieda `DataDecrypter`, ktorá sa postará o dešifrovanie samotných správ. Tieto dve triedy sú abstraktné, a ich implementácia závisí od konkrétneho šifrovacieho algoritmu. Na ich implementáciu bola použitá knižnica BouncyCastle [9].

Po prijatí správy *Server Hello* sa prechádza do stavu `Negotiation`. V tomto stave automat zotrvá po dobu, kým nedostane správu *Client Key Exchange*. Pomocou triedy `KeyDecrypter`, sa dešifruje hodnota *pre-master secret*. Z tejto hodnoty sa pomocou PRF popísanej v RFC 2246 [10] pre TLS v1.0 a TLS v1.1, a v RFC 5246 [2] pre TLS v1.2 získa tzv. *key material*. Pomocou tohto materiálu sa nainicializujú HMAC a symetrická šifra v triede `DataDecrypter`.

Medzi prechodom z inicializačných stavov do stavov, ktoré dešifrujú komunikáciu, je potrebné dešifrovať taktiež správu *Finished*. To je dôležité pri použití prúdových šifier, napr. RC4, ktorých vnútorný stav sa mení (de)šifrovaním každého bytu. *Finished* správa je šifrovaná, jediné čo vieme zistiť po prijatí je, že patrí do Handshake protokolu. Vieme však, že je odosielaná bezprostredne po správe *Change Cipher Spec*. Na to sa spolieha aj algoritmus.

Po tomto kroku algoritmus prechádza do stavu `Intermezzo`. Tento stav vykonáva rovnaké kroky

ako stav `Data Exchange`. Ak by automat prechádzal zo stavu `Negotiation` priamo do stavu `Data Exchange`, cyklus v ktorom je stavový automat uzavretý (a metóda `NewMessage()`) by mohol skončiť bez dešifrovania správy. Zároveň tento stav rieši situáciu ak sa do konverzácie dostane nejaká nedešifrovateľná správa. Teda automat sa neprepne do stavu `Data Exchange`, kým sa nepodarí dešifrovať nejakú správu.

Po prechode do stavu `Data Exchange` sa cyklus automatu skončí. Stav však ostane zachovaný, a pri ďalšom volaní metódy `NewMessage()` sa vykoná cyklus iba raz a jeho výstupom bude dešifrovaná správa. Automat však môže prejsť do iného stavu aj keď je už v `Data Exchange`. Napríklad v prípade SSL Alert protokolu, či zmene parametrov relácie.

Pri parsovaní SSL/TLS správy modul zisťuje koľko bytov SSL/TLS správa obsahuje. Tu sa môže vyskytnúť situácia kedy v reassemblovanom PDU nie je dostatok bytov, respektíve je ich viac. Popisovaný modul túto situáciu rieši tak, že skontroluje ohlásenú dĺžku SSL/TLS správy a porovná ju s množstvom bytov, ktoré dostane zo správy. Ak ich nie je dosť, dáta uloží do dočasného poľa (`ContinuationData`) a dáta z nasledujúceho paketu sa potom prilepia za ne. Podobne je vyriešené ak je dát v správe viac ako je treba, s tým rozdielom, že dáta z pôvodnej správy sa orežú a prebytočné dáta sa po prijatí nasledujúcej správy prilepia pred ňu.

5. Testovanie

Testovanie implementovaného riešenia prebehlo pomocou Unit Testov.

Priebeh SSL/TLS komunikácie sa dá rozdeliť na niekoľko testovateľných častí, ktoré preveria správne dešifrovanie SSL/TLS relácie:

- **Výmena kľúčov**. SSL/TLS protokol môže použiť na výmenu kľúčov rôzne algoritmy asymetrickej kryptografie. Pre každý implementovaný algoritmus je vytvorený osobitný test. V rámci testu sa triede implementujúcej daný algoritmus pošle správa *Client Key Exchange* a preverí sa správnosť získaného kľúča na základe informácií z Wireshark-u.
- **Generovanie kľúča symetrickej kryptografie**. Kľúč symetrickej kryptografie sa generuje v dvoch krokoch, v oboch sa ale používa rovnaká funkcia (PRF). Implementácia tejto funkcie sa mení podľa verzie protokolu.
- **Dešifrovanie správy**. Testovanie samotnej implementácie šifrovacieho algoritmu nemá zmysel, keďže ako implementácia týchto šifrovacích algoritmov bola využitá externá knižnica. Má ale zmysel otestovať, že sa implementovanými krokmi skutočne dostaneme k dešifrovanej správe. Na overenie správnosti implementácie triedy `PDUdecrypter` sa otestuje dešifrovanie jednej relácie SSL/TLS. Overí sa počet dešifrovaných správ a vybratá časť dešifrovaného prúdu dát. Tento postup sa použije pre všetky implementované symetrické šifry.

Vstupné a referenčné hodnoty testov boli získané z debugovacích výstupov Wireshark-u, konkrétne modulu *SSL dissector*. Vo Wireshark-u sa tento modul stará práve o dešifrovanie SSL/TLS relácií. Testovacie dáta[2] boli nazbierané využitím znalostí z kapitoly 3.

Úspešné testy zobrazuje Tabuľka 1.

Tabuľka 1. Úspešné testy modulu PDUDecrypter

Popis testu
Test chovania PRF metódy v TLS v1.0 a TLS v1.1
Test chovania PRF metódy v TLS v1.2
Získanie hodnoty *pre-master secret* zo správy *Client Key Exchange* - RSA
Test dešifrovania - AES v CBC režime
Test dešifrovania - RC4

Neúspešné testy zobrazuje Tabuľka 2.

Tabuľka 2. Neúspešné testy modulu PDUDecrypter

Popis testu	**Dôvod zlyhania**
Test dešifrovania - AES v režime GCM	Obmedzenie knižnice BouncyCastle (využiť inú knižnicu)

6. Záver

Táto práca sa zaoberala TLS/SSL komunikáciou a jej dešifrovaním pre použitie v projekte Netfox. V práci bol navrhnutý modul dešifrovania, ktorý bol následne implementovaný a aktuálne je testovaný. Naimplementované sú niektoré symetrické šifry a asymetrický algoritmus RSA, ktorý sa používa na výmenu kľúčov.

Ďalším pokračovaním je implementácia ostatných verzií protokolu (napr. DTLS) a šifrovacích algoritmov. Zahrnutie autentizačnej časti TLS/SSL handshaku pre budúcu implementáciu statickej verzie Difiie-Hellman.

Výstupom je kompletný modul dešifrovania použitý v projekte Netfox. V prípade prístupu k privátnemu kľúču serveru dovoľuje Netfoxu extrakciu a analýzu šifrovaných dát. Keďže tento privátny kľúč nie je bežne dostupný, je možné pôvodnú SSL/TLS reláciu ukončiť na nejakom bode medzi klientom a serverom (proxy), nahradiť pôvodný privátny kľúč vlastným, a

[2] http://1drv.ms/1Cmg3es

Obrázok 4. Možnosti nástroja Netfox.Framework s implementovaným modulom.

nadviazať novú SSL/TLS reláciu od proxy ku klientovy. Popisované nové možnosti spojené s týmto modulom sú na obrázku 4.

Tento modul rozširuje možnosti rodiny nástrojov Netfox v oblasti forenznej analýzy počítačových sietí a zákonných odposluchov.

Literatúra

[1] Jan Pluskal. Framework for captured network communication. diplomová práce, FIT VUT v Brně, Brno, 2014.

[2] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, RFC Editor, August 2008.

[3] A. Freier, P. Karlton, and P. Kochar. The Secure Socket Layer (SSL) Protocol Version 3.0. RFC 6101, RFC Editor, August 2011.

[4] William Stallings. *Cryptography and Network Security*. Prentice Hall, 1998.

[5] E. Rescorla. Diffie-Hellman Key Agreement Method. RFC 2631, RFC Editor, June 1991.

[6] S. Davidoff and J. Ham. *Network Forensics*. Prentice Hall, 2012.

[7] Daniel Roethlisberger. Sslsplit - transparent and scalable ssl/tls interception. https://www.roe.ch/SSLsplit, 2015 cit. 2015-03-23.

[8] Moxie Marlinspike. Moxie marlinspike » software » sslstrip. http://www.thoughtcrime.org/software/sslstrip/, 2012 cit. 2015-03-23.

[9] The legion of the bouncy castle. https://www.bouncycastle.org/, 2013 cit. 2015-03-23.

[10] T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246, RFC Editor, January 1999.