

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2064

**ANALIZA I KATEGORIZACIJA MREŽNOG PROMETA
ŠIFRIRANOG KORIŠTENJEM PROTOKOLA TLS**

KORISNIČKE UPUTE

Marko Plantić

Zagreb, veljača 2020.

Sadržaj

1.1.	Uvod	3
1.2.	Pokretanje skripti	3
1.3.	Korišćenje skripte za analizu client hello poruka	4
1.4.	Korišćenje skripte za analizu TCP SYN paketa	6
1.5.	Korišćenje skripte za analizu šifriranih podataka	7
1.6.	Završne napomene	10

1.1. Uvod

Ovim korisničkim uputama je opisan rad skripta za analizu šifriranog mrežnog prometa u sklopu diplomskog rada na temu "Analiza i kategorizacija mrežnog prometa šifriranog korištenjem protokola TLS". Postoji sveukupno tri skripta koje analiziraju snimljeni mrežni promet. One su:

1. `client_hello_analyzer.js` – analizira client hello poruke snimljenih TLS paketa za identifikaciju web preglednika iz liste skupa sigurnosnih algoritama
2. `tcp_syn_analyzer.js` – analizira TCP SYN pakete za TLS promet, koristi se za dobivanje jedinstvenih vrijednosti TTL i veličine prozora
3. `web_analyzer.js` – analizira snimljen TLS promet preko Wireshark-a te uspoređuje sa snimljenim prometom u web pregledniku

1.2. Pokretanje skripti

Za pokretanje gore navedenih skripti su potrebni sljedeći alati:

- git - <https://git-scm.com/downloads>
- nodejs - <https://nodejs.org/en/download/>

Koraci za preuzimanje i pokretanje skripti:

1. `$ git clone https://github.com/Kuyss/diplomski_rad.git`
2. `$ cd diplomski_rad`
3. `$ npm install`
4. `$ cd src/scripts`
5. `$ node [naziv_skripte]` – pokretanje skripte

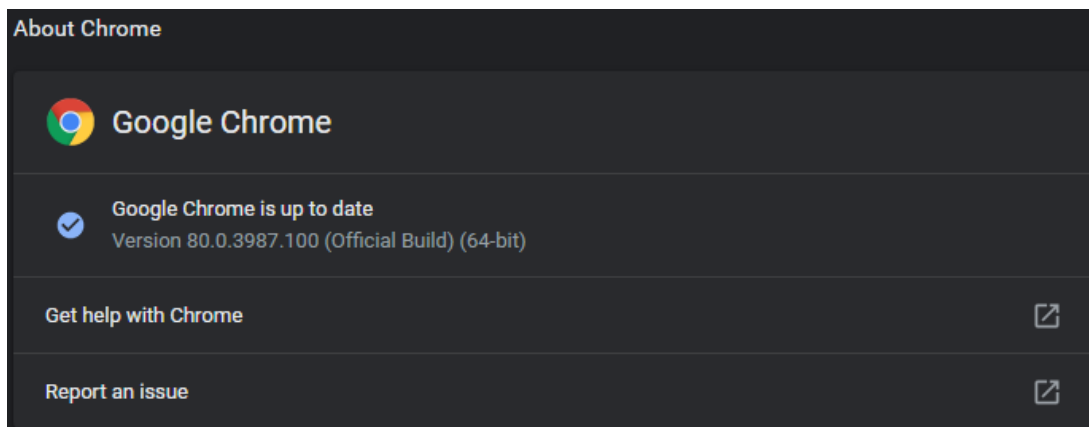
1.3. Korištenje skripte za analizu client hello poruka

Za analizu client hello poruka TLS prometa se koristi skripta `client_hello_analyzer.js`. Skripta se pokreće na sljedeći način:

- `$ node client_hello_analyzer.js`

Ova skripta kao ulazne podatke sadrži sljedeće datoteke:

- `tls_clientHello.pcap` – pcap datoteka koja sadrži sve client hello poruke tijekom uspostave sigurne TLS veze. TLS client hello poruke se dobivaju korištenjem filtera `tls.handshake.type == 1` u Wireshark display filterima.
- `cipher_suite_db.json` – JSON datoteka koja sadrži bazu čiji elementi su verzija web preglednika i lista skupa sigurnosnih algoritama u heksadekadskom formatu. Baza se dobiva tako da se u web pregledniku pristupi nekoj web stranici sa HTTPS vezom te se u Wireshark-u snimi client hello poruka. Slika 1 prikazuje verziju korištenog preglednika Google Chrome. Do nje dolazimo preko options > Help > About Google Chrome. Zatim se pronade client hello poruka za taj preglednik u Wireshark-u te se odabere u drugom prozoru Transport Layer Protocol > TLS Record Layer > Handshake Protocol > Cipher Suites. Zatim se desnim klikom na Cipher Suites odabere izbornik Copy > ...as a Hex Stream. Slika 2. prikazuje ovaj korak. Zadnji korak je dodavanje vezije preglednika i liste skupa sigurnosnih algoritama u JSON datoteku koja predstavlja bazu podataka (Slika 3).



Slika 1. Korištena verzija Google Chrome web preglednika

tls.handshake.type == 1

No.	Time	Source	Destination	Protocol	Length	Info
41746	471.338685	192.168.11.106	13.107.42.12	TLSv1.2	230	Client Hello
41808	474.168807	192.168.11.106	52.166.78.97	TLSv1.2	571	Client Hello
42018	475.987828	192.168.11.106	52.166.78.97	TLSv1.2	571	Client Hello
42022	475.991121	192.168.11.106	52.166.78.97	TLSv1.2	571	Client Hello
42025	475.991993	192.168.11.106	52.166.78.97	TLSv1.2	571	Client Hello
42086	476.040239	192.168.11.106	52.166.78.97	TLSv1.2	571	Client Hello
42337	476.524537	192.168.11.106	52.174.224.26	TLSv1.2	571	Client Hello
42343	476.564256	192.168.11.106	52.174.224.26	TLSv1.2	571	Client Hello
42370	476.631869	192.168.11.106	52.174.224.26	TLSv1.2	571	Client Hello

> Frame 42022: 571 bytes on wire (4568 bits), 571 bytes captured (4568 bits) on interface \Device\NPF_{9AFB77E4-AAB0-4...}

> Ethernet II, Src: Micro-St_d0:58:64 (4c:cc:6a:d0:58:64), Dst: Tp-LinkT_ad:85:92 (ac:84:c6:ad:85:92)

> Internet Protocol Version 4, Src: 192.168.11.106, Dst: 52.166.78.97

> Transmission Control Protocol, Src Port: 52011, Dst Port: 443, Seq: 1, Ack: 1, Len: 517

▼ Transport Layer Security

▼ TLSv1.2 Record Layer: Handshake Protocol: Client Hello

Content Type: Handshake (22)

Version: TLS 1.0 (0x0301)

Length: 512

▼ Handshake Protocol: Client Hello

Handshake Type: Client Hello (1)

Length: 508

Version: TLS 1.2 (0x0303)

> Random: 461948ab38602a9a528e5575ea1db81db4c12191659736e0...

Session ID Length: 32

Session ID: 6a17000036f8d8a8e7583079211f7d789fcb48ef0af0a684...

Cipher Suites Length: 34

> Cipher Suites (17 suites)

Compression Methods Length: 2

> Compression Methods (1 method)

Extensions Length: 401

> Extension: Reserved (0)

> Extension: server_name

> Extension: extended_master_secret

> Extension: renegotiation_info

> Extension: supported_groups

> Extension: ec_point_formats

> Extension: session_ticket

> Extension: application_protocols

> Extension: status_request

> Extension: signature_algorithms

> Extension: signed_certificate_timestamp

> Extension: key_share

> Extension: psk_key_exchange_modes

> Extension: supported_versions

> Extension: compress_certificate

> Extension: Reserved (0)

> Extension: padding

0080 ee 5c 00 22 7a 7a 13 01 13

0090 c0 2c c0 30 cc a9 cc a8 c0

00a0 00 2f 00 35 00 0a 01 00

00b0 00 0f 00 0d 00 0a 77 77

00c0 72 00 17 00 00 ff 01 00 01

List of cipher suites supported by client

Expand Subtrees

Collapse Subtrees

Expand All

Collapse All

Apply as Column Ctrl+Shift+I

Apply as Filter

Prepare as Filter

Conversation Filter

Colorize with Filter

Follow

Copy

Show Packet Bytes... Ctrl+Shift+O

Export Packet Bytes... Ctrl+Shift+X

Wiki Protocol Page

Filter Field Reference

Protocol Preferences

Decode As...

Go to Linked Packet

Show Linked Packet in New Window

All Visible Items

All Visible Selected Tree Items

Description

Field Name

Value

As Filter

Copy Bytes as Hex + ASCII Dump

...as Hex Dump

...as Printable Text

...as a Hex Stream

...as Raw Binary

Slika 2. Kopiranje liste skupa sigurnosnih algoritama

```

{
  "browser": "Google Chrome 79.0.3945.130",
  "cipherSuites": "130113031302c02bc02fcc9cca8c02cc030c00ac009c013c01400330039002f0035000a"
},
{
  "browser": "Mozilla Firefox 72.0.1",
  "cipherSuites": "6a6a130113021303c02bc02fc02cc030cca9cca8c013c014009c009d002f0035000a"
},
{
  "browser": "Microsoft Edge 44.18362.449.0",
  "cipherSuites": "c02cc02bc030c02fc024c023c028c027c00ac009c014c013009d009c003d003c0035002f000a"
}

```

Slika 3. Baza verzije web preglednika i liste skupa sigurnosnih algoritama

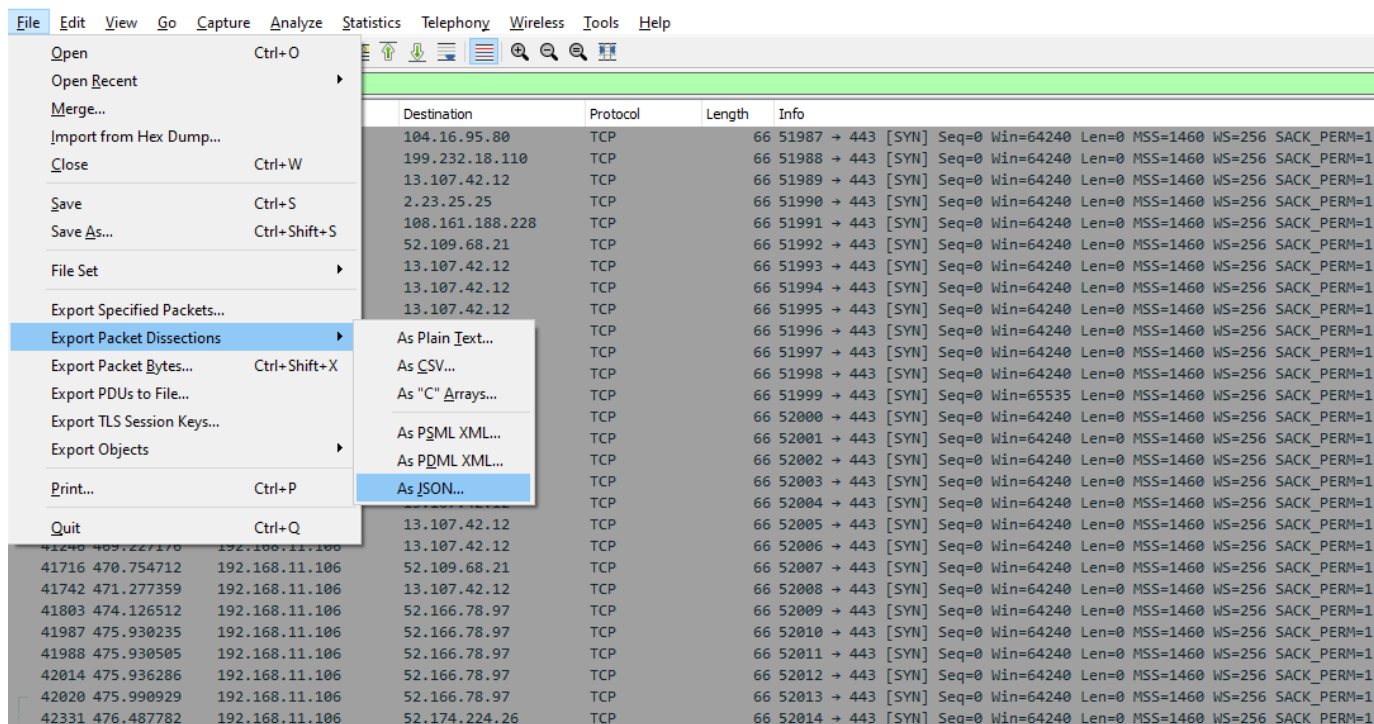
Skripta obrađuje svaki paket iz `tls_clientHello.pcap` datoteke, određuje listu skupa sigurnosnih algoritama te ju uspoređuje sa bazom spremljenom u `cipher_suite_db.json`. Rezultat usporedbe su verzije web preglednika te broj stvorenih veza od svakog preglednika te se printa u konzolu.

1.4. Korištenje skripte za analizu TCP SYN paketa

Za analizu TCP SYN paketa se koristi skripta `tcp_syn_analyzer.js`. Pokreće se sljedećom naredbom:

- `$ node tcp_syn_analyzer.js`

Kao ulaz sadrži datoteku `tcp_syn.json` koja predstavlja snimljene TCP SYN pakete izvezen u JSON formatu. Korišteni filter za prikaz TCP SYN paketa je `tcp.flags.syn == 1 && tcp.flags.ack == 0`. Slika 4 prikazuje način stvaranja JSON formata snimljenih TCP SYN paketa. Zatim skripta za svaki paket odabire TTL i veličinu prozora te uspoređuje sa vrijednostima spremljenim u listi. Ako već postoje iste vrijednosti, ne ubacuje se u listu. Ako ne postoje, ubacuje se u listu. Rezultat je datoteka `tcp_syn_db.json` u kojoj su spremljene različite vrijednosti TTL i veličine prozora.



Slika 4. Izvoz TCP SYN paketa u JSON format

1.5. Korištenje skripte za analizu šifriranih podataka

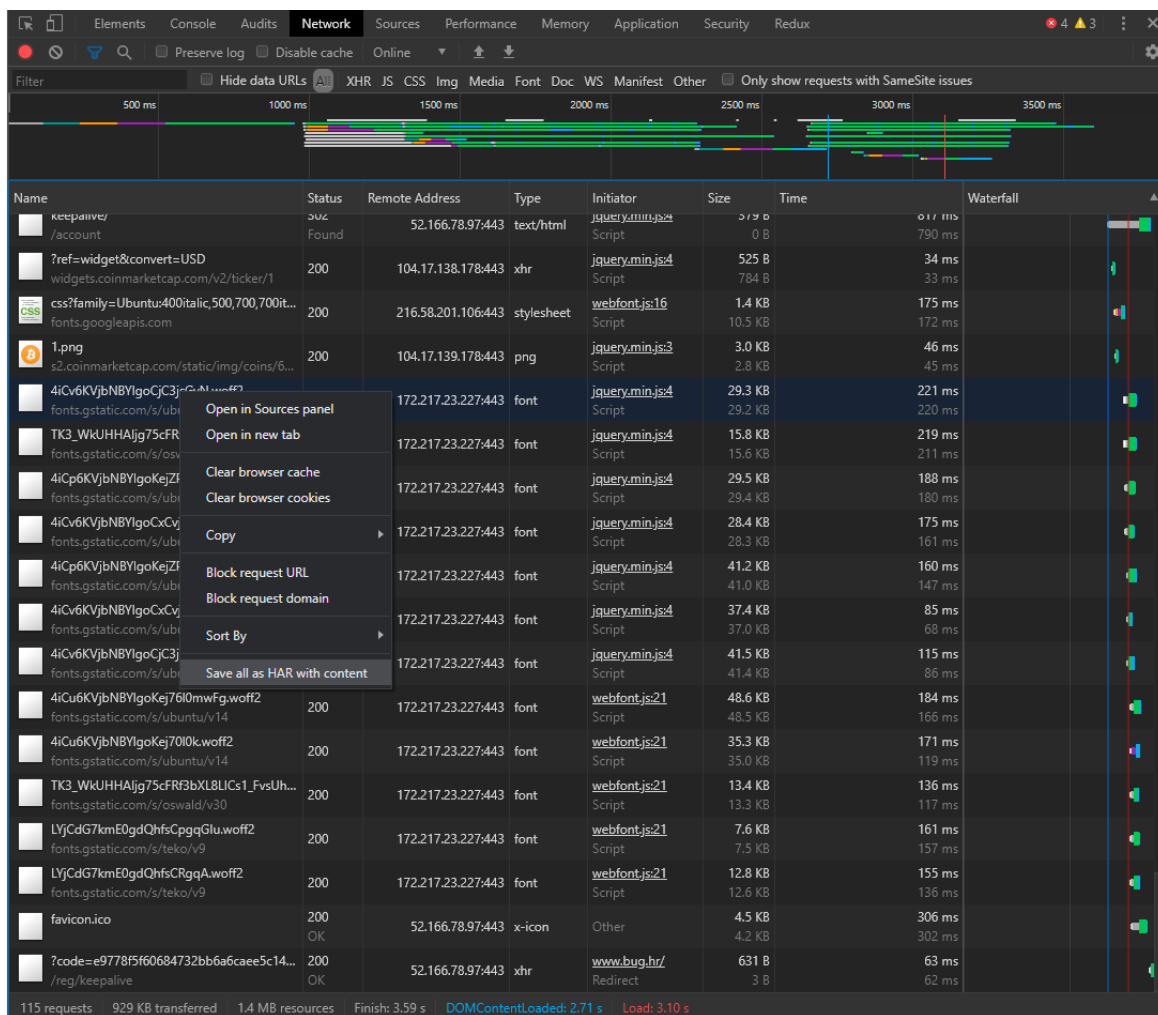
Za analizu šifriranih podataka se koristi skripta `web_analyzer.js`. Njome analiziramo i identificiramo kojem web članku bug.hr portala je korisnik kojeg snimamo pristupio.

Pokreće se na dva načina:

- `$ node web_analyzer.js calc` – obrađuje spremljene har datoteke prometa snimljenog web preglednikom. Prvo se pozicioniramo na članak koji želimo snimati. Zatim pritisnemo F12 čime otvaramo Chrome DevTools. Zatim odaberemo refresh ikonu > desni klik mišem > Empty Cache and Hard Reload (Slika 5). Ovime omogućujemo da se svi resursi iz poslužitelja povuku, bez obzira jesu li bili spremljeni (keširani) u priručni spremnik. Nakon što se sav promet učita, odaberemo Network tab u DevTools-ima, desni klik mišem > save all as HAR with content (slika 6).



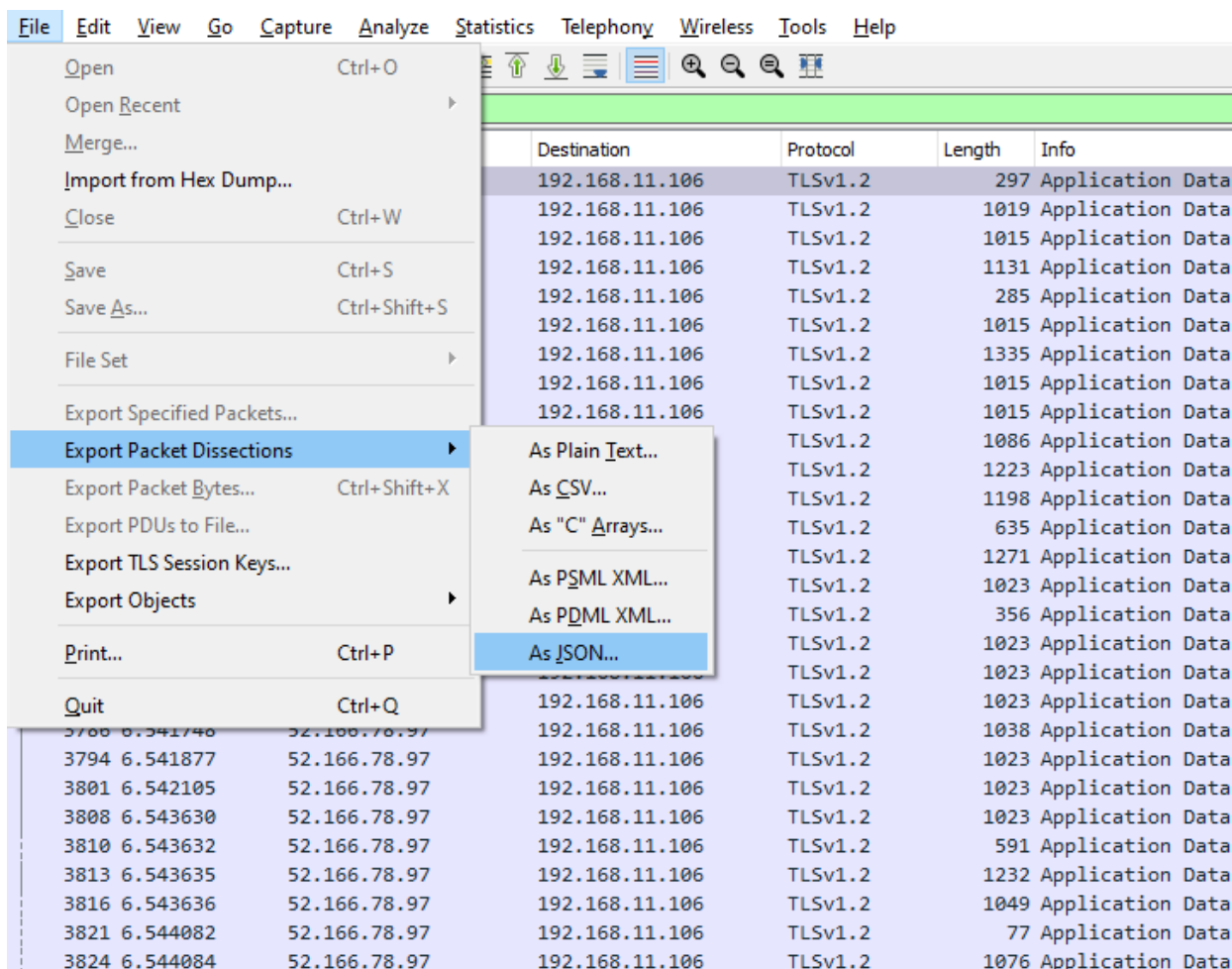
Slika 5. Odabir načina učitavanja web članka



Slika 6. Spremanje snimljenog prometa u HAR datoteku

Važno je napomenuti da je potrebno spremiti har datoteke u formatu data{num}.har, gdje je num broj od 1 do 10.

- \$ node web_analyzer.js get – za ovaj način rada je potrebno snimiti promet preko Wireshark-a korištenjem sljedećeg filtra za svaku posjetu web članku: *ip.src == 52.166.78.97 && tls.app_data*. Time dobijemo odgovor od bug.hr poslužitelja sa svim vraćenim resursima. Sljedeći korak je spremiti snimljeni promet za pojedini članak u JSON formatu te nazvati datoteke capture{num}.json, gdje je num broj od 1 do 10 (slika 7). Zatim se za svaki snimljeni članak uspoređuje veličina vraćenih resursa sa bazom dobivenom prvim korakom te ispisuje najvjerojatnije pristupljeni članak.



Slika 7. Snimljeni promet za pojedini web članak bug.hr portala

1.6. Završne napomene

U GitHub repozitoriju diplomskog rada ne postoje datoteke sa snimljenim TCP SYN paketima jer zauzimaju ogromnu količinu prostora (500MB). Također postoji skripta crawler.js kojom sam pokušao automatizirati proces dohvata članka sa bug.hr portala i spremanja har datoteka. Na kraju sam odusato od tog načina analize veličine vraćenih resursa za svaki članak jer se koristio web preglednik iz komandne linije. Veličine vraćenih resursa web preglednika Google Chrome i onog iz komandne linije se nisu podudarale. Pretpostavio sam da je razlika zbog različitih implementacija preglednika te bi se možda odabirom drugog preglednika iz komandne linije mogli dobiti konzistentniji rezultati.