Reversible Covert Channels Over ~~HTTPS~~ TLS

May 14, 2024

Supervised By: Professor Doctor Jürgen Schönwälder

By: Joshua Law

Motivation

- Few literature about implementing Reversible Data Hiding methods onto Network Covert Channels
- Previously works only implemented with IPv4 packets
- Understanding the mechanisms and characteristics essential for developing effective defensive strategies and countermeasures.
- Knowing how covert channels operate lets us learn how to detect and mitigate it

Objective

- Explore the Reversible Covert Channel Methodology proposed previously, applied over packets in a secure data stream
- Develop a prototype model that can simulate the data exchange between client and server with a reversible covert channel built between as intermediate network nodes
- Measure how many bytes of message we can send and how effective our method is

Covert Channels Types

- Classic Covert Channels
  - Storage Covert Channels
    - Using storage systems (eg. File names) to communicate
  - Timing Covert Channels
    - Receiver observing response times (eg. Paging rates)
- Network Covert Channels
  - Uses the network environment as medium

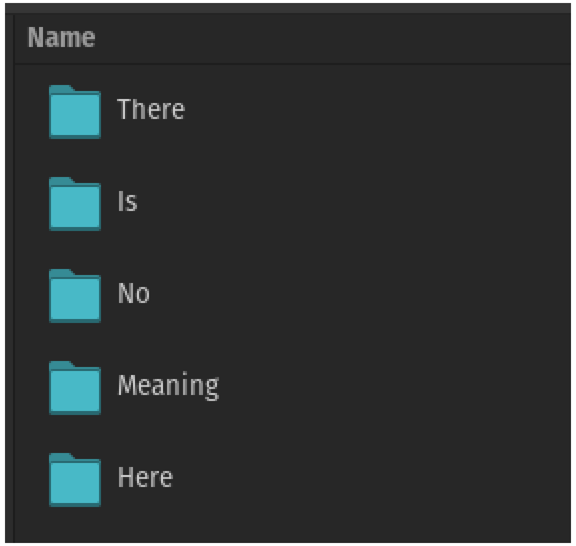


Figure: Potential Covert Channel

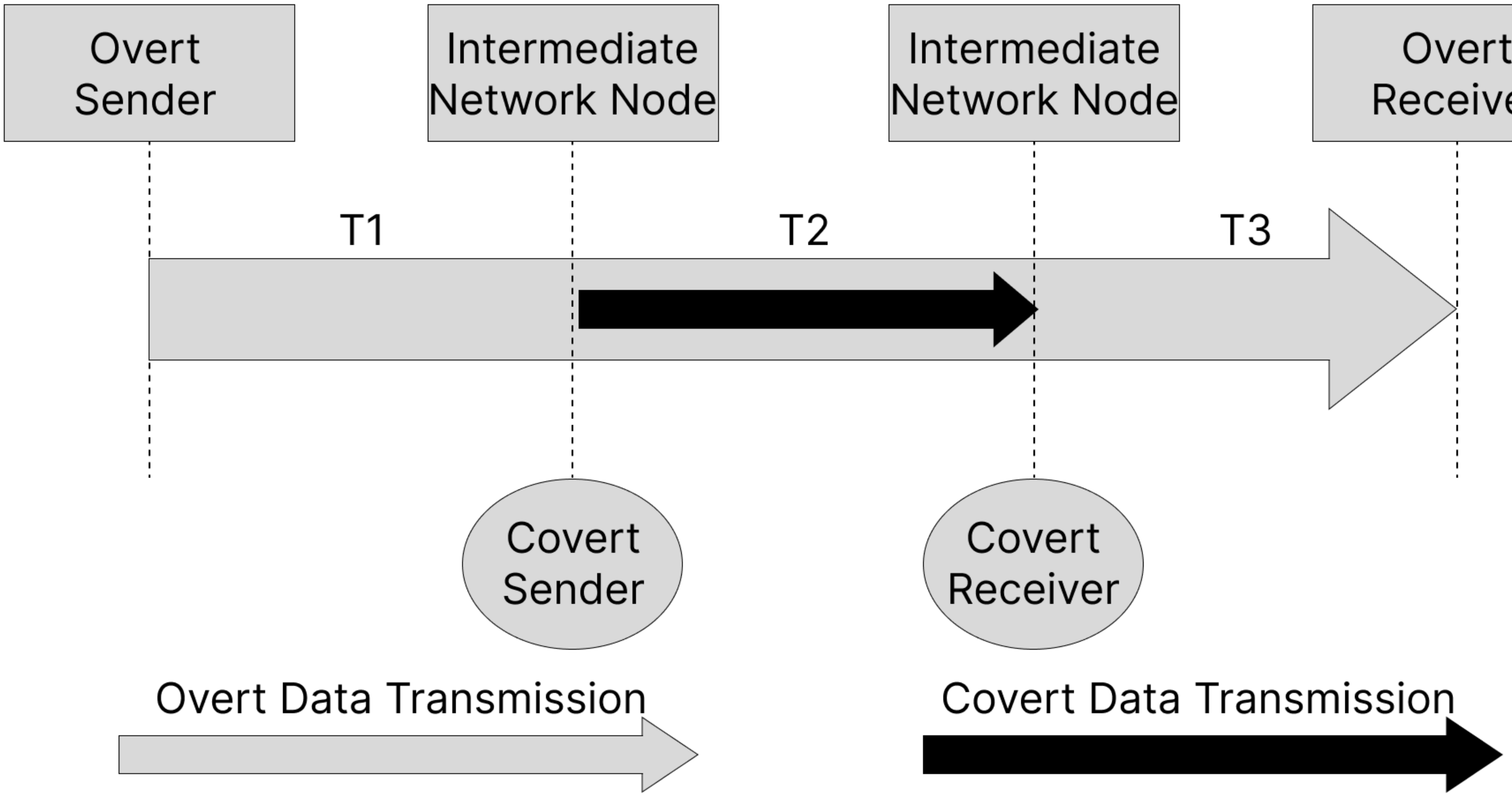
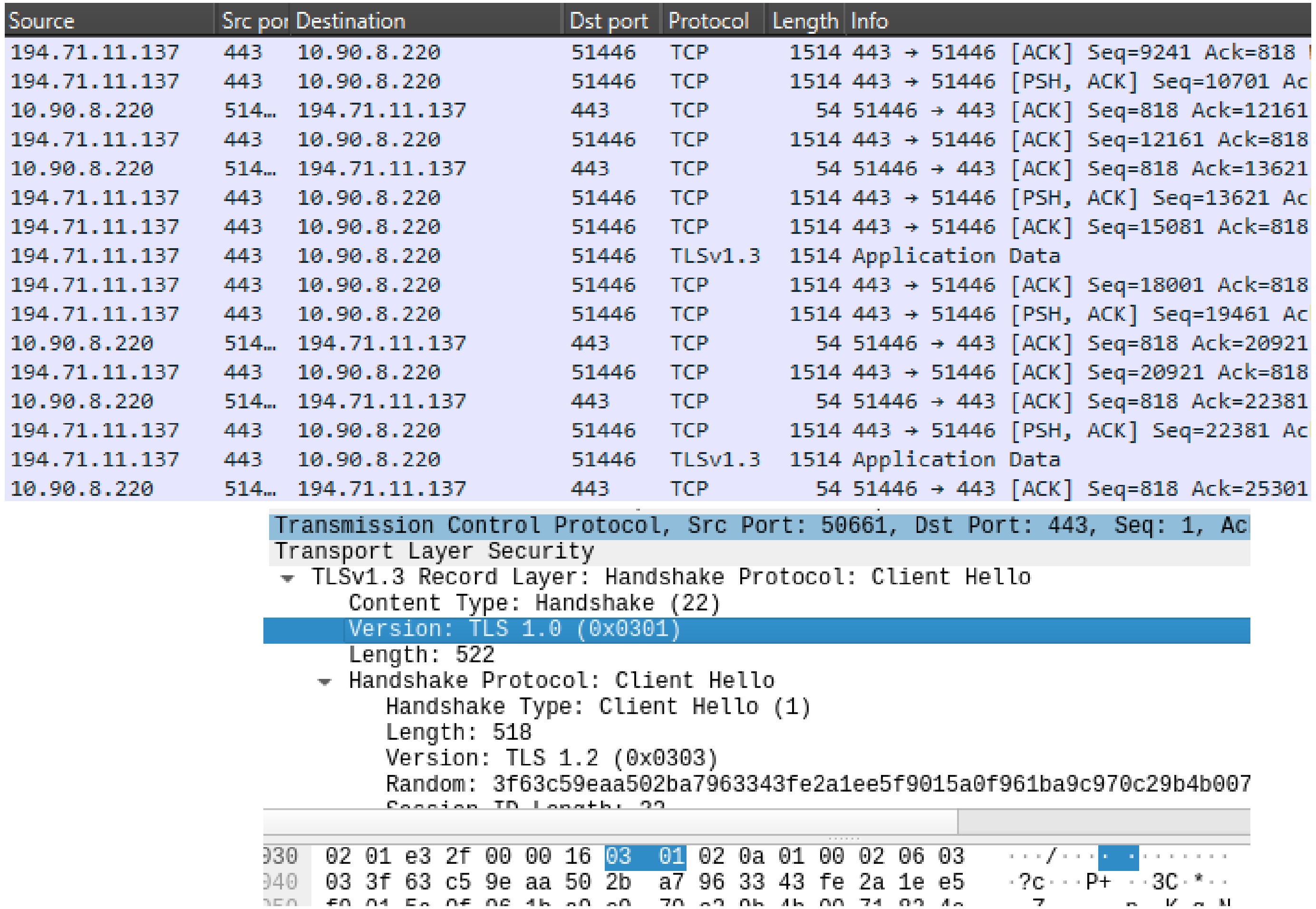


Figure: Data transmission in a Network Covert Channel Structure

Main Technologies Used

- Wireshark
- Python Libraries:
  - Scapy
  - Mininet
  - libNetfilter Queue
- Linux iptables



Source	Src por	Destination	Dst port	Protocol	Length	Info
194.71.11.137	443	10.90.8.220	51446	TCP	1514	443 → 51446 [ACK] Seq=9241 Ack=818
194.71.11.137	443	10.90.8.220	51446	TCP	1514	443 → 51446 [PSH, ACK] Seq=10701 Ac
10.90.8.220	514…	194.71.11.137	443	TCP	54	51446 → 443 [ACK] Seq=818 Ack=12161
194.71.11.137	443	10.90.8.220	51446	TCP	1514	443 → 51446 [ACK] Seq=12161 Ack=818
10.90.8.220	514…	194.71.11.137	443	TCP	54	51446 → 443 [ACK] Seq=818 Ack=13621
194.71.11.137	443	10.90.8.220	51446	TCP	1514	443 → 51446 [PSH, ACK] Seq=13621 Ac
194.71.11.137	443	10.90.8.220	51446	TCP	1514	443 → 51446 [ACK] Seq=15081 Ack=818
194.71.11.137	443	10.90.8.220	51446	TLSv1.3	1514	Application Data
194.71.11.137	443	10.90.8.220	51446	TCP	1514	443 → 51446 [ACK] Seq=18001 Ack=818
194.71.11.137	443	10.90.8.220	51446	TCP	1514	443 → 51446 [PSH, ACK] Seq=19461 Ac
10.90.8.220	514…	194.71.11.137	443	TCP	54	51446 → 443 [ACK] Seq=818 Ack=20921
194.71.11.137	443	10.90.8.220	51446	TCP	1514	443 → 51446 [ACK] Seq=20921 Ack=818
10.90.8.220	514…	194.71.11.137	443	TCP	54	51446 → 443 [ACK] Seq=818 Ack=22381
194.71.11.137	443	10.90.8.220	51446	TCP	1514	443 → 51446 [PSH, ACK] Seq=22381 Ac
194.71.11.137	443	10.90.8.220	51446	TLSv1.3	1514	Application Data
10.90.8.220	514…	194.71.11.137	443	TCP	54	51446 → 443 [ACK] Seq=818 Ack=25301

```
Transmission Control Protocol, Src Port: 50661, Dst Port: 443, Seq: 1, Ac
Transport Layer Security
  TLSv1.3 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 522
    Handshake Protocol: Client Hello
      Handshake Type: Client Hello (1)
      Length: 518
      Version: TLS 1.2 (0x0303)
      Random: 3f63c59eaa502ba7963343fe2a1ee5f9015a0f961ba9c970c29b4b007

030  02 01 e3 2f 00 00 16 03  01 02 0a 01 00 02 06 03   ···/···· ········
040  03 3f 63 c5 9e aa 50 2b  a7 96 33 43 fe 2a 1e e5   ·?c···P+ ··3C·*··
```

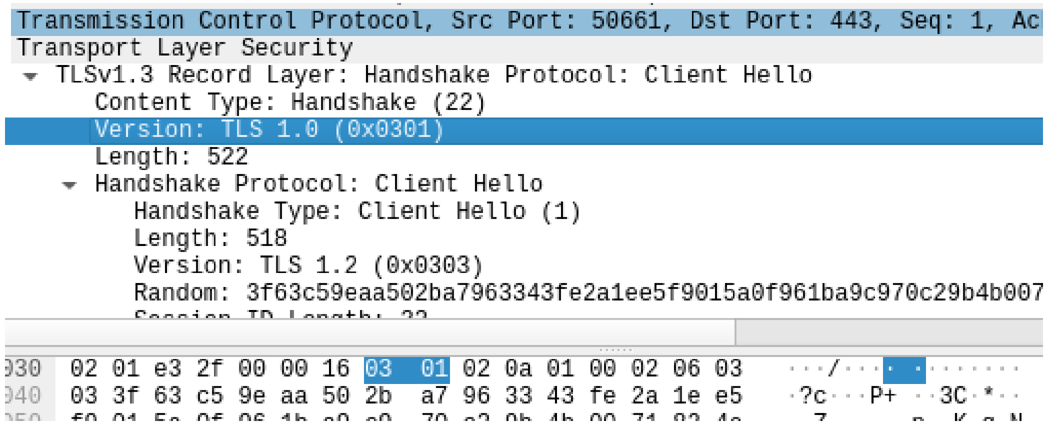
Figure: Wireshark example usage

Transport Layer Security (TLS) Records

- TLS serves as an additional security layer on top of the IP/TCP transport protocols.
- TLS record protocol secures application data using the keys created during Handshake.
- Version number follows the pattern: 0x0301 for TLS 1.0 and 0x0303 for TLS 1.2 , etc.

Table: TLS record content type bytes

Hexadecimal	Decimal	Protocol
0x14	20	Application Data
0x16	22	Handshake
0x17	23	Change Cipher Spec



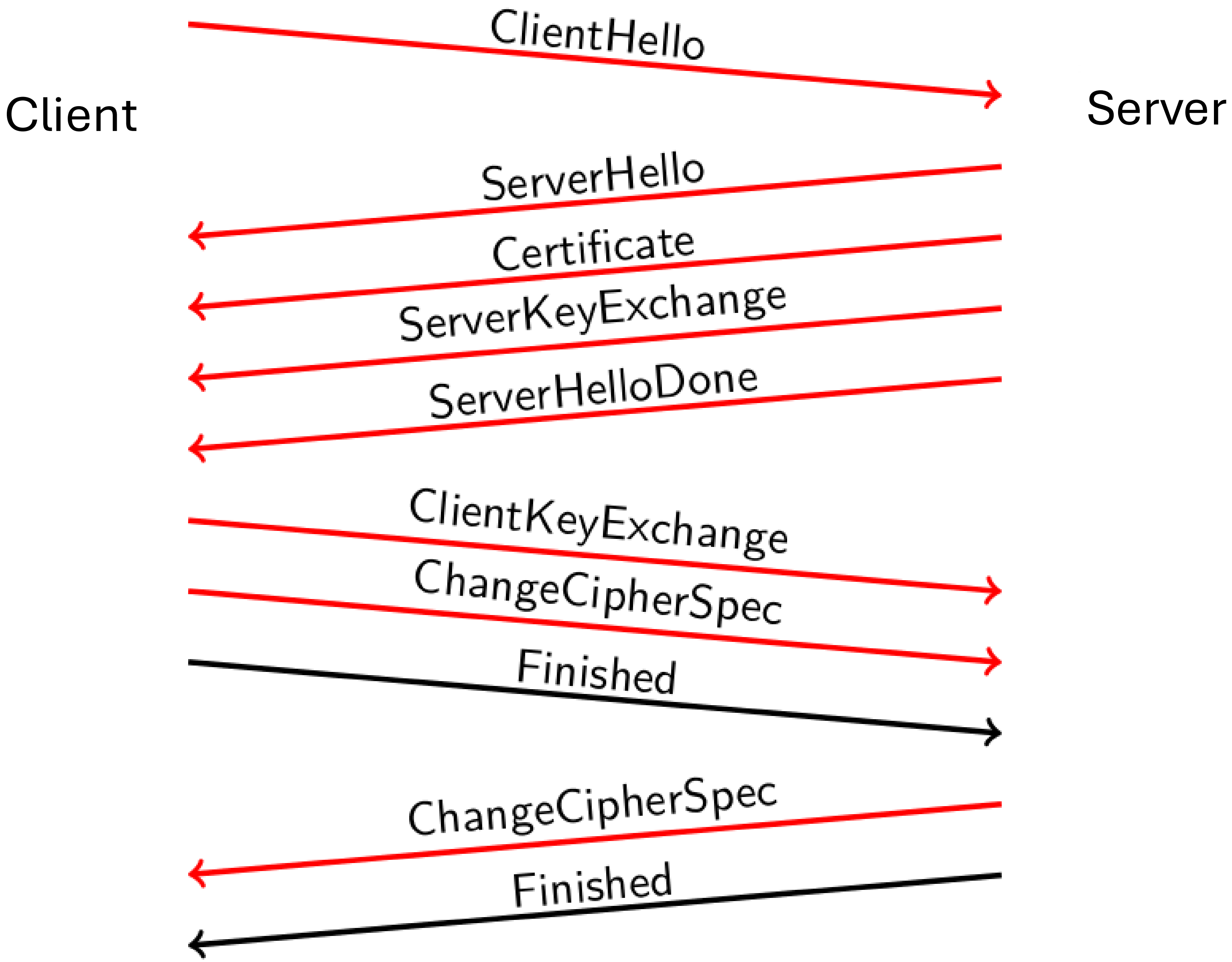


Figure: TLS overview

Description of Investigation

- Define the Mininet topology suitable for the simulation
- Route packets using iptable rules
- Implementation using Scapy and nfqueue to intercept and modify packet
- Identifying the TLS packet within the TCP stream
- Parsing for the TLS record version number
- Modifying the version number at Intermediate network node 1
- Receiving and restoring bytes at Intermediate network node 2
- Observe with Wireshark

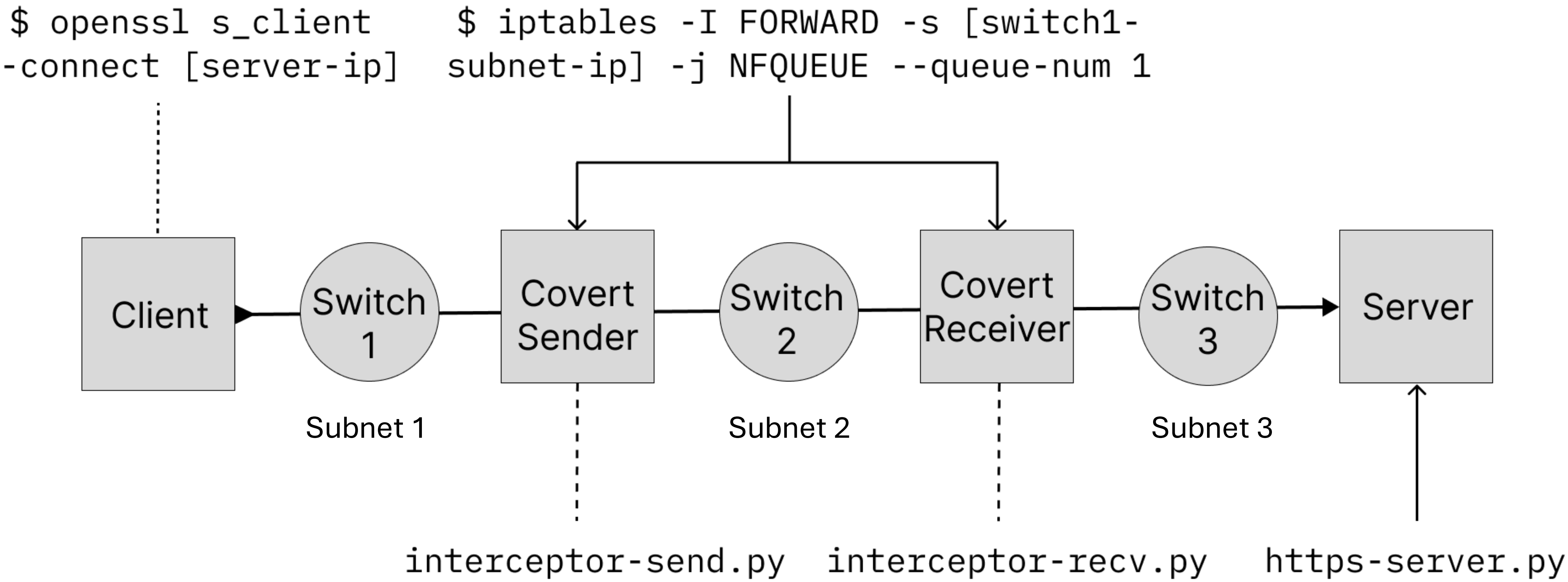


Figure: Experiment Mininet implementation

Investigation Results

- Suitable implementation using nfqueues and scapy allow packet to be routed to userspace and achieve an information exchange between the two middle points
- Using only TLS Record Version Number, we can only get 2 bytes per TLS packet, minimum 4 bytes unidirectional.
- Change in response speed with the reversible covert channel averages to an increase in 0.4 ms, limitation of implementation.

Conclusion

- A working prototype shows that we can effectively intercept and modify the TLS record version packet, avoiding detection if we restore the packets before being received at the endpoint.
- TLS records are proven to be suitable candidates for constructing reversible covert channels
- The speed of the data transmission is not affected enough such that it is noticeably slower

Further Work

- Countermeasures against cyber attacks such as the Reversible Covert Channel here.
- Improving the implementation of the Reversible Covert Channel methodology presented in terms of scalability and efficiency.
- Implementation of the Reversible Covert Channel that addresses the possibility of a security implementation between the covert sender and receiver.

References

- Krzysztof Cabaj et al. "The New Threats of Information Hiding: The Road Ahead". In: IT Professional 20.3 (2018), pp. 31–39. DOI: 10.1109/MITP.2018.032501746.
- Secdev. *Scapy TLS notebooks*. URL: https://github.com/secdev/scapy/blob/master/doc/notebooks/tls/notebook2_tls_protected.ipynb.
- Wojciech Mazurczyk et al. Information Hiding in Communication Networks: Fundamentals, Mechanisms, and Applications. Mar. 2016. ISBN: 978-1-118-86169-1.
- Wojciech Mazurczyk et al. "Towards Reversible Storage Network Covert Channels". In: Aug. 2019. ISBN: 978-1-4503-7164-3. DOI: 10.1145/3339252.3341493.
- Gerald Combs. Wireshark. Version 4.2.4. URL: www.wireshark.org.