Secure Network Management

CNSM Part I - CNF Project (TFTP Proxy) project report – Group 4

Hochschüle München

Sommersemester 2024

Authored by:

Celina Linnerblom Persson Semester ID: 020916 Affiliated university [Erasmus Exchange]: University of Borås, Faculty of Librarianship, Information, Education and IT Matriculation Number: 35133224 Home adress: Kavaljersgatan 20, 422 48 Hisings Backa, Gothenburg, Sweden
Beata Jacobsson Semester ID: 021235 Affiliated university [Erasmus Exchange]: University of Borås, Faculty of Librarianship, Information, Education and IT Matriculation Number: 34772424 Home adress: Mannerfelts väg 16B, 504 31 Borås, Sweden

Table of Contents

1. Summary

The following project aim to implement and analyze a Virtual LAN, including developing a proxy application in order to manipulate exchanged messages between the client and the server using TFTP.

The objectives of this projects highlights learning insights of how to create a virtual network using virtual machines and to create a understanding of how traffic runs between different host within the network. Additionally, gain insights on how to intercept traffic, manipulate it as well as analyze behavior of the protocols between client and server. This project requries understanding of python programming language and it´s key libraries in context of networking, which is expecting to be slightly challanging due to no previous experience of the python language. Furthermore this assignment aims to increase our abilities to dissect network traffic through the Wireshark Analyzer.

Our expectations for this project is to develop a solid understanding of the computer networking fundamentals and gain practical experience in implementing code for managing faulty situations. By completing this project we aim to enhance our overall knowledge and skills in the area of networking.

2. Network setup

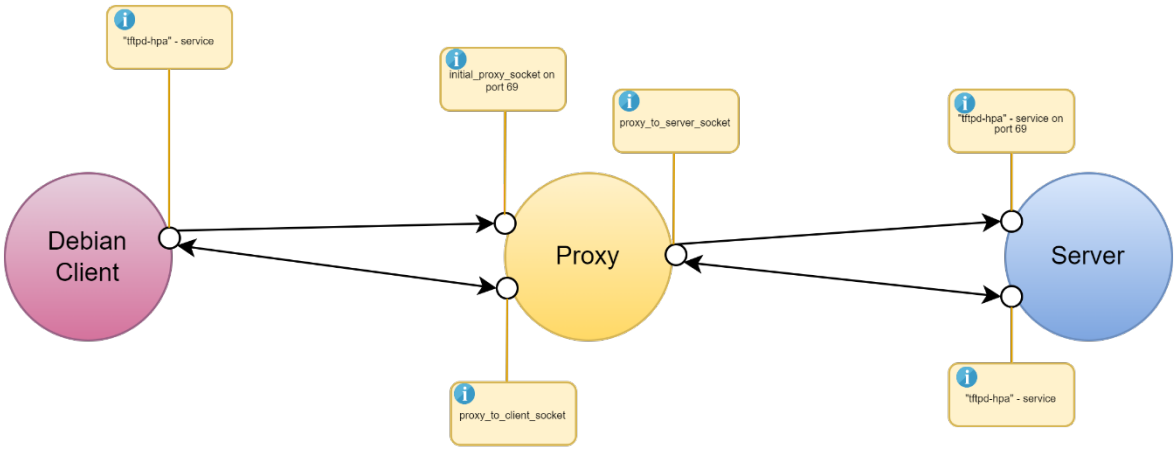


Figure 1: Overview of the network environment setup.

2.1 Configuration description

The project is developed in the open-source tool Virtualbox. The used virtual machines are pre-configured and used as a virtual laboratory to orchastrate the following faulty situations. The Client (Debian) hosts the necessary software packages e.g Visual Studio Code, TFTP client software and ssh, which we have extended in order to complete this assignment. The Proxy is used to intercept the traffic while the Server runs the Server-Application in order to communicate with it.

TFTP (Trivial File Transfer Protocol) is a file transfer protocol that uses UDP (User Datagram Protocol) on port 69 for transferring files. TFTP ensures reliablility at the application layer to compensate the unreliablity of the UDP tranport layer protocol. This through mechanisms such as numbering all messages, acknowledgements messages and timers.

3. Analysis/Result section of simulated faulty situations

3.1 Increase Payload size of the data packet to over 512 bytes in RRQ/WRQ traffic

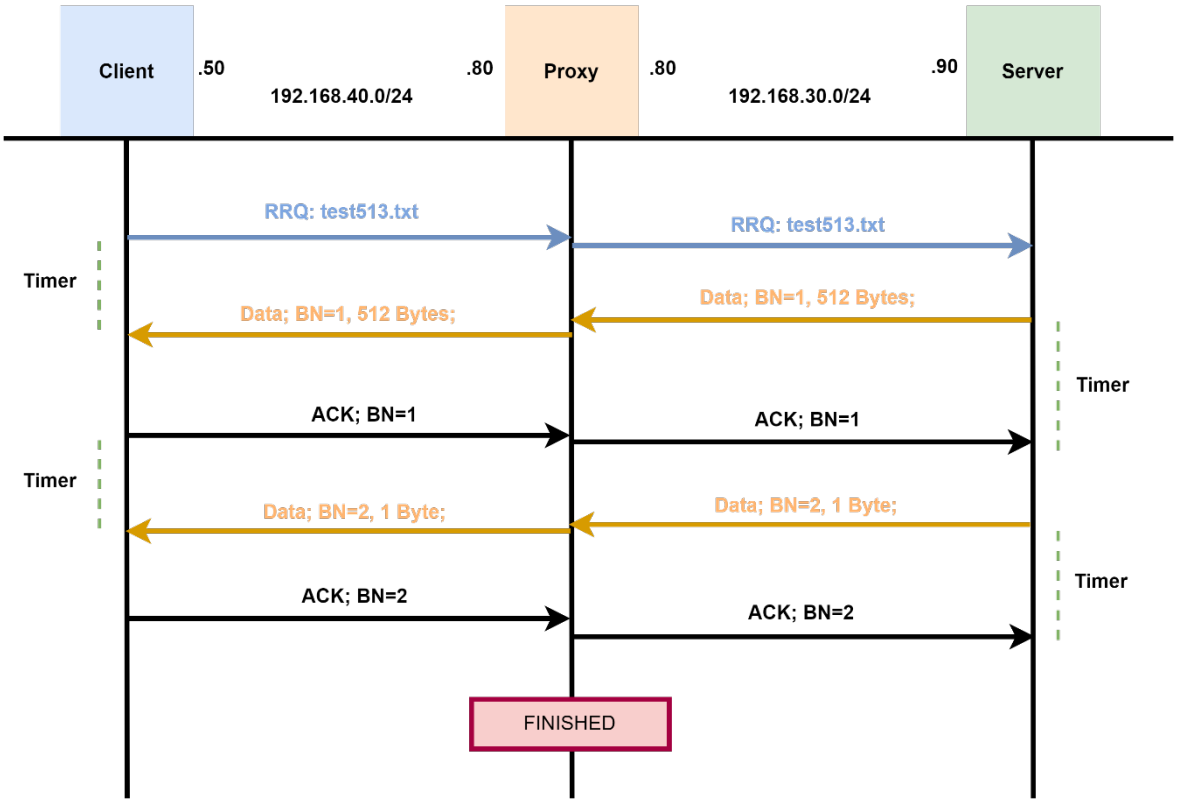


Figure 2: Flow diagram of faulty situation 1

tftp

No.	Time	Source	Destination	VLAN	Protocol	Src Port	Dst Port	VLAN ID	Info
175	29.396572619	192.168.40.50	192.168.40.80		TFTP	44083	69		Read Request, File: test513.txt,
176	29.400482294	192.168.30.80	192.168.30.90	30	TFTP	60706	69	30	Read Request, File: test513.txt,
179	29.416430267	192.168.30.90	192.168.30.80	30	TFTP	39331	60706	30	Data Packet, Block: 1
184	29.421024463	192.168.40.80	192.168.40.50		TFTP	35371	44083		Data Packet, Block: 1
185	29.421506589	192.168.40.50	192.168.40.80		TFTP	44083	35371		Acknowledgement, Block: 1
188	29.423561067	192.168.30.80	192.168.30.90	30	TFTP	60706	39331	30	Acknowledgement, Block: 1
191	29.431522961	192.168.30.90	192.168.30.80	30	TFTP	39331	60706	30	Data Packet, Block: 2 (last)
192	29.436201016	192.168.40.80	192.168.40.50		TFTP	35371	44083		Data Packet, Block: 2 (last)
195	29.438939073	192.168.40.50	192.168.40.80		TFTP	44083	35371		Acknowledgement, Block: 2
196	29.440612027	192.168.30.80	192.168.30.90	30	TFTP	60706	39331	30	Acknowledgement, Block: 2

Figure 3: Captured traffic during increased size of datapacket

Analysis:

In this scenario proxy is employed to modify TFTP packets. The proxy increases the payload size of the data packets beyond the standard 512 bytes limit during RRQ/WRQ traffic. The TFTP server consistently reject packets with payloads larger than 512 bytes. This outcome underscores the protocol's strict standards about how big the packets can be. For the network configuration involving TFTP and the proxy setup a client requests and server responses is used. The server responses includes datapackets and acknowledge packets. Two data packets is used where one is 512 bytes and one is 1 bytes. Two acknowledge packets are utilized, one with block number 1 and the other with block number 2. To measure and analyze network traffic in this project, we used Wireshark for capturing and analyzing data packets, along with Python for programming tasks related to handling and processing network comunnications.

3.2 Change BN of ACK packet in RRQ/WRQ traffic

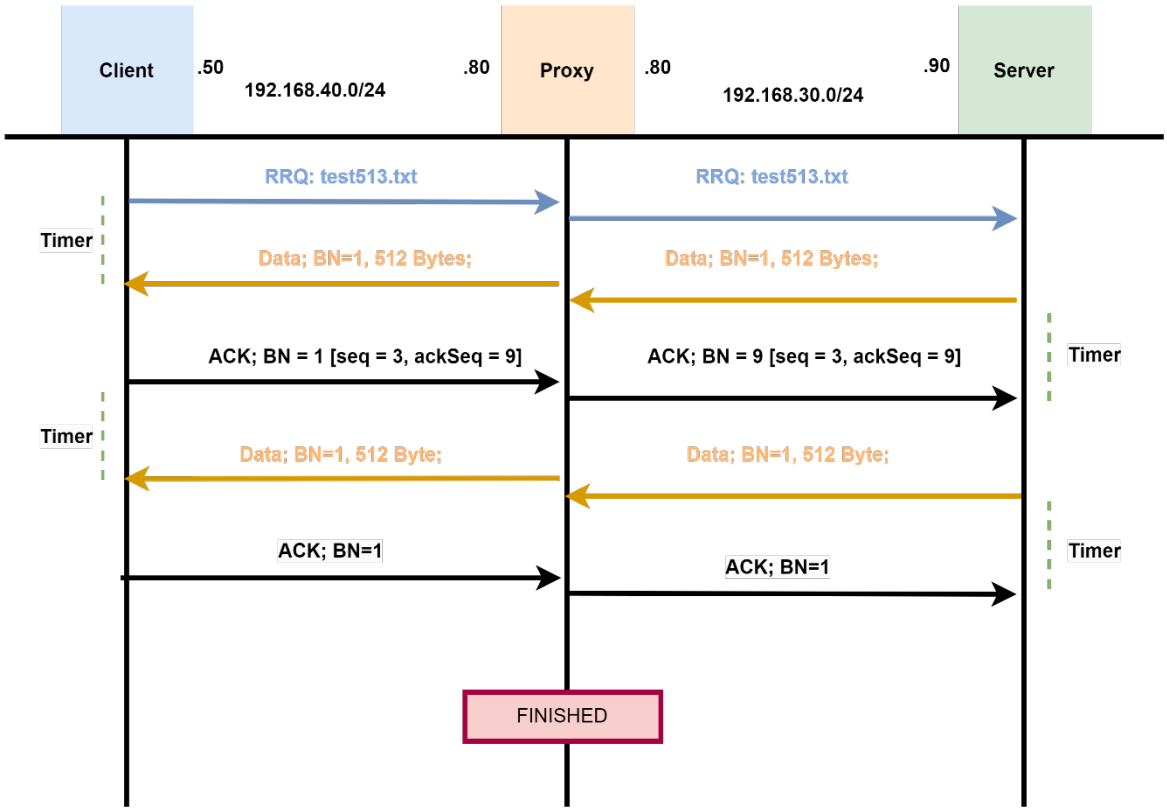


Figure 4: Flow diagram of faulty situation 2

tftp

No.	Time	Source	Destination	VLAN	Protocol	Src Port	Dst Port	VLAN ID	Info
180	29.912557618	192.168.40.50	192.168.40.80		TFTP	55086	69		Read Request, File: test513.txt, Transfer t
181	29.915146575	192.168.30.80	192.168.30.90	30	TFTP	37358	69	30	Read Request, File: test513.txt, Transfer t
184	29.924170521	192.168.30.90	192.168.30.80	30	TFTP	43136	37358	30	Data Packet, Block: 1
185	29.925417673	192.168.40.80	192.168.40.50		TFTP	37222	55086		Data Packet, Block: 1
186	29.926137907	192.168.40.50	192.168.40.80		TFTP	55086	37222		Acknowledgement, Block: 1
189	29.928972271	192.168.30.80	192.168.30.90	30	TFTP	37358	43136	30	Acknowledgement, Block: 9
192	30.927351483	192.168.30.90	192.168.30.80	30	TFTP	43136	37358	30	Data Packet, Block: 1
193	30.929116561	192.168.40.80	192.168.40.50		TFTP	37222	55086		Data Packet, Block: 1
194	30.929355665	192.168.40.50	192.168.40.80		TFTP	55086	37222		Acknowledgement, Block: 1
197	30.932177220	192.168.30.80	192.168.30.90	30	TFTP	37358	43136	30	Acknowledgement, Block: 1
198	30.932177421	192.168.30.90	192.168.30.80	30	TFTP	43136	37358	30	Data Packet, Block: 2 (last)
205	31.934822105	192.168.30.90	192.168.30.80	30	TFTP	43136	37358	30	Data Packet, Block: 2 (last)
206	31.937299376	192.168.30.80	192.168.30.90	30	ICMP	43136	37358	30	Destination unreachable (Port unreachable)
211	35.936078448	192.168.40.50	192.168.40.80		TFTP	55086	37222		Acknowledgement, Block: 1
212	35.937656811	192.168.40.80	192.168.40.50		ICMP	55086	37222		Destination unreachable (Port unreachable)

Figure 5: Captured traffic during a modified blocknumber of first ACK packet

Analysis:

This transmission focuses on altering the blocknumber of the first acknowledgment packet. The Python script utilizes sockets to intercept, forward and modify these packets. Through our setup the proxy successfully intercepts the initial RRQ (get test513.txt) packet from the client and subsequent datapackets from the server. It then manipulates the blocknumber of first ACK packet by setting it to 9 instead of 1. By doing this, the first datapacket is retransmissioned by the server, until the ACK packet of the correct blocknumber is recieved. Thus does the verification process confim proxy to be handeling the packets correctly, ensuring a seemless communication between the client and server despite the intentional modicfication.

3.3 Transmission delay for data packet for 25 seconds in RRQ traffic

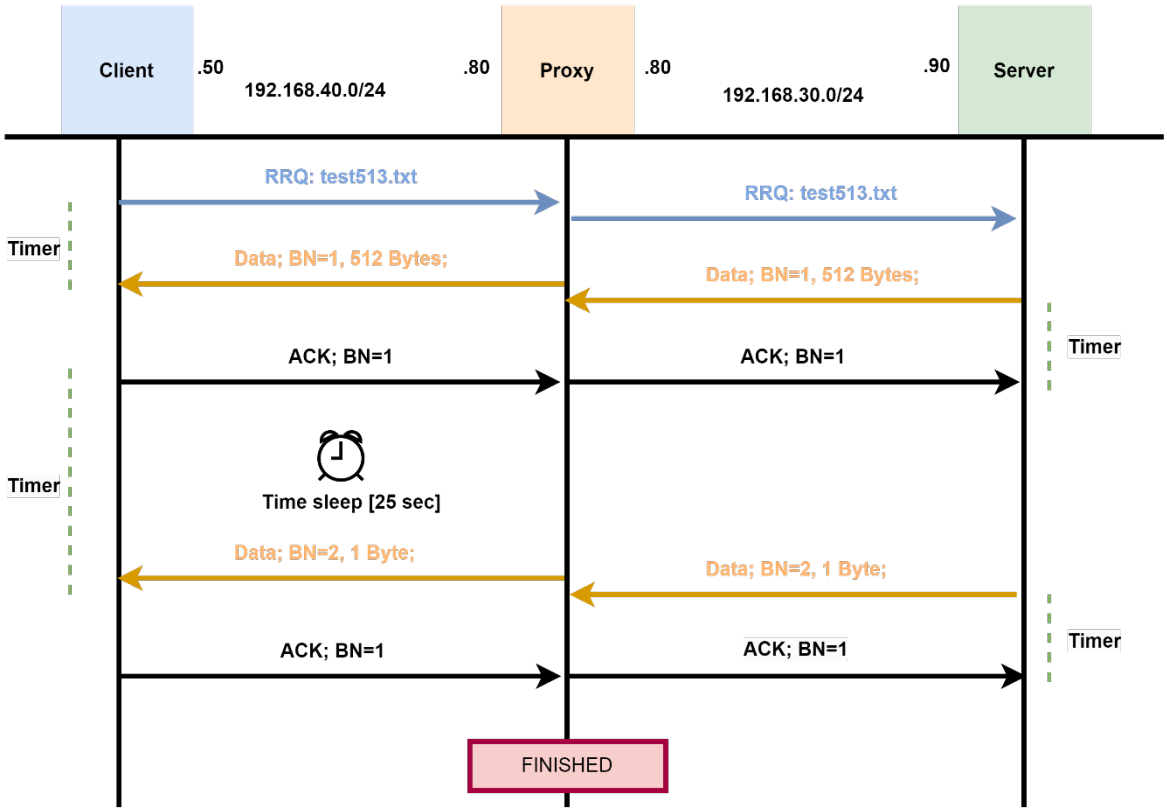


Figure 6: Flow diagram of faulty situation 3

tftp

No.	Time	Source	Destination	VLAN	Protocol	Src Port	Dst Port	VLAN ID	Info
180	26.539989948	192.168.40.50	192.168.40.80		TFTP	49258	69		Read Request, File: test513.txt,
181	26.542256859	192.168.30.80	192.168.30.90	30	TFTP	51891	69	30	Read Request, File: test513.txt,
184	26.548194240	192.168.30.90	192.168.30.80	30	TFTP	37777	51891	30	Data Packet, Block: 1
187	26.551601188	192.168.40.80	192.168.40.50		TFTP	52103	49258		Data Packet, Block: 1
190	26.553537960	192.168.40.50	192.168.40.80		TFTP	49258	52103		Acknowledgement, Block: 1
191	26.556669341	192.168.30.80	192.168.30.90	30	TFTP	51891	37777	30	Acknowledgement, Block: 1
194	26.560634872	192.168.30.90	192.168.30.80	30	TFTP	37777	51891	30	Data Packet, Block: 2 (last)
197	27.565109226	192.168.30.90	192.168.30.80	30	TFTP	37777	51891	30	Data Packet, Block: 2 (last)
198	29.568645782	192.168.30.90	192.168.30.80	30	TFTP	37777	51891	30	Data Packet, Block: 2 (last)
199	31.556859836	192.168.40.50	192.168.40.80		TFTP	49258	52103		Acknowledgement, Block: 1
204	33.576199601	192.168.30.90	192.168.30.80	30	TFTP	37777	51891	30	Data Packet, Block: 2 (last)
205	36.557738057	192.168.40.50	192.168.40.80		TFTP	49258	52103		Acknowledgement, Block: 1
206	41.561137757	192.168.40.50	192.168.40.80		TFTP	49258	52103		Acknowledgement, Block: 1
207	41.587939345	192.168.30.90	192.168.30.80	30	TFTP	37777	51891	30	Data Packet, Block: 2 (last)
208	46.561693457	192.168.40.50	192.168.40.80		TFTP	49258	52103		Acknowledgement, Block: 1
209	51.598111812	192.168.40.80	192.168.40.50		TFTP	52103	49258		Data Packet, Block: 2 (last)
214	51.602091109	192.168.30.80	192.168.30.90	30	TFTP	51891	37777	30	Acknowledgement, Block: 1
223	57.604567955	192.168.30.90	192.168.30.80	30	TFTP	37777	51891	30	Data Packet, Block: 2 (last)
224	57.605827908	192.168.30.80	192.168.30.90	30	ICMP	37777	51891	30	Destination unreachable (Port unr

208	46.561693457	192.168.40.50	192.168.40.80		TFTP	49258	52103		Acknowledgement, Block: 1
209	51.598111812	192.168.40.80	192.168.40.50		TFTP	52103	49258		Data Packet, Block: 2 (last)
214	51.602091109	192.168.30.80	192.168.30.90	30	TFTP	51891	37777	30	Acknowledgement, Block: 1
223	57.604567955	192.168.30.90	192.168.30.80	30	TFTP	37777	51891	30	Data Packet, Block: 2 (last)
224	57.605827908	192.168.30.80	192.168.30.90	30	ICMP	37777	51891	30	Destination unreachable (Port


```
▾ [Timestamps]
    [Time since first frame: 25.046510624 seconds]
    [Time since previous frame: 5.036418355 seconds]
```

Figure 7: Captured traffic during transmission delay of 25 secounds

Analysis:

This setup included deliberated delay of 25 secounds that is introduced for transmitting data packets from the proxy to the clieint during RRQ operations. After receving a second data packet containing 1 byte from the server, a delay of 25 seconds was implemented using time.sleep(25) before forwarding this packet to the client. The client's acknowledgement for the second data block is recivied and forwarded back to the server. This delay was strategically incorporated to simulate scenarios where network responses may be delayed, affecting the timing of data packets delivery during file transer.

3.4 Request a file over 512 bytes and delay ACK packet for 25 seconds in RRQ traffic

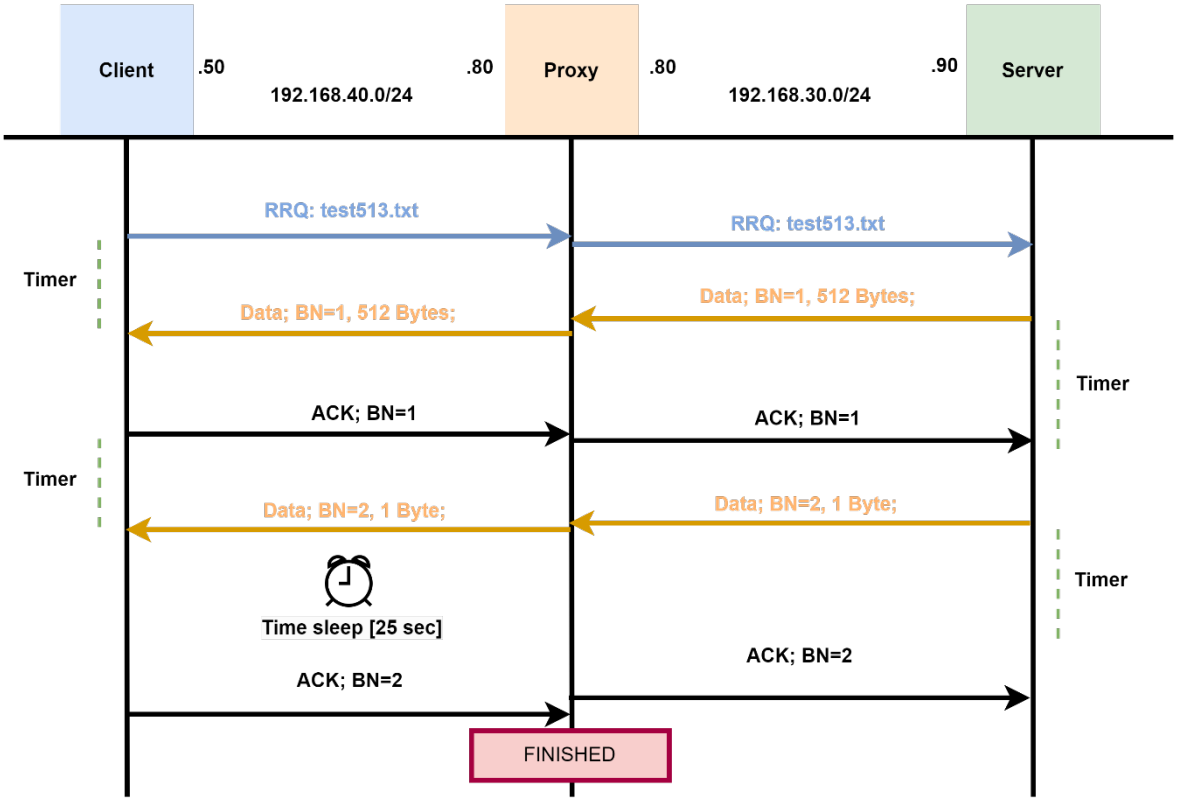


Figure 8: Flow diagram of faulty situation 4

tftp

No.	Time	Source	Destination	VLAN	Protocol	Src Port	Dst Port	VLAN ID	Info
10	8.452135305	192.168.40.50	192.168.40.80		TFTP	37343	69		Read Request, File: test513.txt,
11	8.453916756	192.168.30.80	192.168.30.90	30	TFTP	51735	69	30	Read Request, File: test513.txt,
14	8.456953195	192.168.30.90	192.168.30.80	30	TFTP	43847	51735	30	Data Packet, Block: 1
15	8.457569868	192.168.40.80	192.168.40.50		TFTP	49396	37343		Data Packet, Block: 1
16	8.457812078	192.168.40.50	192.168.40.80		TFTP	37343	49396		Acknowledgement, Block: 1
23	8.461694833	192.168.30.80	192.168.30.90	30	TFTP	51735	43847	30	Acknowledgement, Block: 1
25	8.463831329	192.168.30.90	192.168.30.80	30	TFTP	43847	51735	30	Data Packet, Block: 2 (last)
26	8.463831358	192.168.40.80	192.168.40.50		TFTP	49396	37343		Data Packet, Block: 2 (last)
28	8.464163991	192.168.40.50	192.168.40.80		TFTP	37343	49396		Acknowledgement, Block: 2
34	9.464534809	192.168.30.90	192.168.30.80	30	TFTP	43847	51735	30	Data Packet, Block: 2 (last)
36	11.465374201	192.168.30.90	192.168.30.80	30	TFTP	43847	51735	30	Data Packet, Block: 2 (last)
45	15.467321210	192.168.30.90	192.168.30.80	30	TFTP	43847	51735	30	Data Packet, Block: 2 (last)
52	23.471036234	192.168.30.90	192.168.30.80	30	TFTP	43847	51735	30	Data Packet, Block: 2 (last)
53	33.475597686	192.168.30.80	192.168.30.90	30	TFTP	51735	43847	30	Acknowledgement, Block: 2

Figure 9: Captured traffic during a large file request and delayed ACK packet

Analysis:

This scenario requests a file larger than 512 bytes and intentionally delays the ACK-packet which follows, to ensure the file size excees 512 bytes two datapackets are sent. When the request from the client is made, the first data-packet and it´s corresponding acknowledgment is forwarded without delay. Although, before the forwarding of the second ACK, the proxy introduces a deliberate delay of 25 seconds before forwarding this ACK-packet to the server. This is done by *time module* and the function *sleep*. When the ACK-packet is delayed the server will retransmitt the data-packet repeatedly

until the recieves the expected acknowledgement. This behavior is due to the timeout and retransmission mechanism built into the TFTP protocol, ensuring reliable data transfer.

3.5 Replace a data packet with an error packet in WRQ traffic

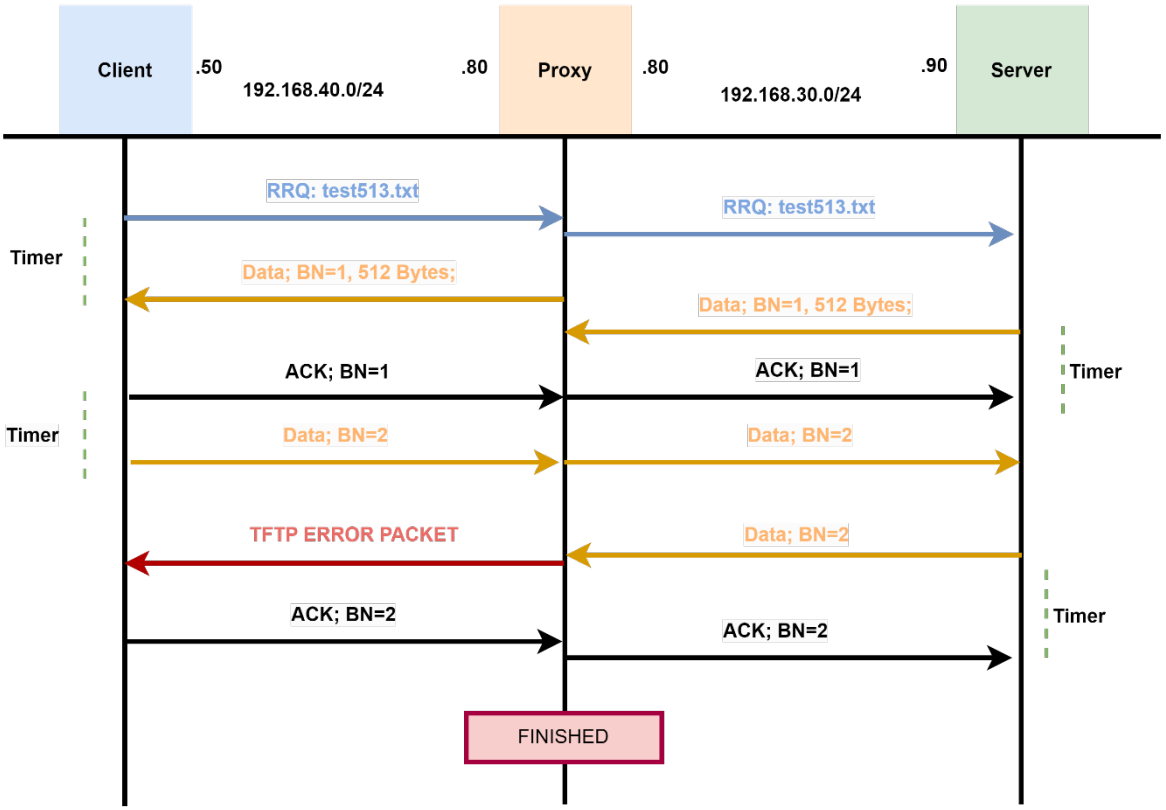
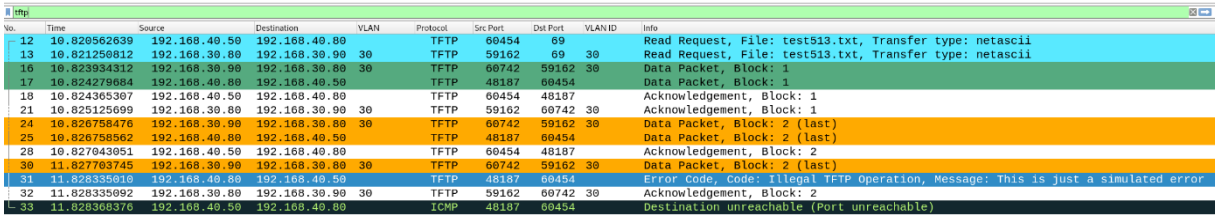


Figure 10: Flow diagram of faulty situation 5



No.	Time	Source	Destination	VLAN	Protocol	Src Port	Dst Port	VLAN ID	Info
12	10.820562639	192.168.40.50	192.168.40.80		TFTP	60454	69		Read Request, File: test513.txt, Transfer type: netascii
13	10.821250812	192.168.30.80	192.168.30.90	30	TFTP	59162	69	30	Read Request, File: test513.txt, Transfer type: netascii
16	10.823934312	192.168.30.90	192.168.30.80	30	TFTP	60742	59162	30	Data Packet, Block: 1
17	10.824279684	192.168.40.80	192.168.40.50		TFTP	48187	60454		Data Packet, Block: 1
18	10.824365307	192.168.40.50	192.168.40.80		TFTP	60454	48187		Acknowledgement, Block: 1
21	10.825125699	192.168.30.80	192.168.30.90	30	TFTP	59162	60742	30	Acknowledgement, Block: 1
24	10.826758476	192.168.30.90	192.168.30.80	30	TFTP	60742	59162	30	Data Packet, Block: 2 (last)
25	10.826758562	192.168.40.80	192.168.40.50		TFTP	48187	60454		Data Packet, Block: 2 (last)
28	10.827043051	192.168.40.50	192.168.40.80		TFTP	60454	48187		Acknowledgement, Block: 2
30	11.827703745	192.168.30.90	192.168.30.80	30	TFTP	60742	59162	30	Data Packet, Block: 2 (last)
31	11.828335010	192.168.40.80	192.168.40.50		TFTP	48187	60454		Error Code, Code: Illegal TFTP Operation, Message: This is just a simulated error
32	11.828335092	192.168.30.80	192.168.30.90	30	TFTP	59162	60742	30	Acknowledgement, Block: 2
33	11.828360376	192.168.40.50	192.168.40.80		ICMP	48187	60454		Destination unreachable (Port unreachable)

Figure 11: Captured traffic with recieved ERROR packet

Analysis:

For this scenario a function is implemented to intercepts and manipulates TFTP packets during a WRQ operation. Instead of forwarding the final data packet receivied from the server, the function generates an error packet. The error packet includes an opcode indicating an error, an error code and an error message. This validation confirmed that the server correctly interpreted and handled the error packet. The error packet is forwarded to the client instead of the expected data packet, simulating an error scenario during file transfer. Lastly the function waits for an aknowledge packet from the client to acknowledge the error, which is forwarded to the server.

```
        2 bytes  2 bytes        string    1 byte
        -----------------------------------------
ERROR | 05    |  ErrorCode |   ErrMsg   |   0  |
        -----------------------------------------
```

(Sollins, K.R., 1992)

```python
# Create an error packet instead of forwarding the last data packet
error_opcode = OPCode.ERROR.value.to_bytes(2, 'big')
error_code = (4).to_bytes(2, 'big')
error_msg = b'This is just a simulated error'
error_packet = error_opcode + error_code + error_msg + b'\x00'

# Forward the error packet to the client instead of the last data packet
proxy.forward(proxy_to_client_socket, client_address, error_packet)
```

Figure 12: Python code for building an Error-packet according to it´s message format

3.6 Request a file larger than 512 bytes. Change BN of second data packet in RRQ traffic

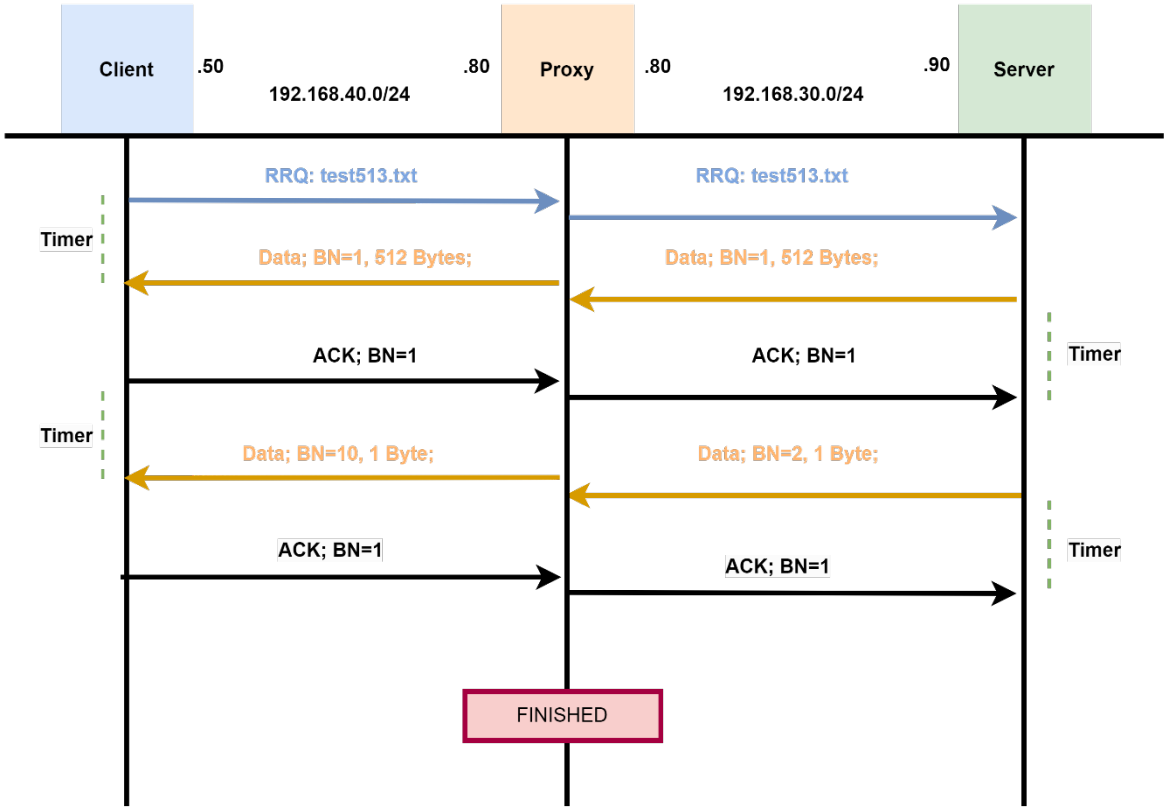


Figure 13: Flow diagram of faulty situation 6

tftp

No.	Time	Source	Destination	VLAN	Protocol	Src Port	Dst Port	VLAN ID	Info
53	16.199136945	192.168.40.50	192.168.40.80		TFTP	51673	69		Read Request, File: test513.txt
54	16.201727031	192.168.30.80	192.168.30.90	30	TFTP	50544	69	30	Read Request, File: test513.txt
57	16.208979296	192.168.30.90	192.168.30.80	30	TFTP	38300	50544	30	Data Packet, Block: 1
58	16.213947240	192.168.40.80	192.168.40.50		TFTP	38033	51673		Data Packet, Block: 1
61	16.214401852	192.168.40.50	192.168.40.80		TFTP	51673	38033		Acknowledgement, Block: 1
63	16.216722951	192.168.30.80	192.168.30.90	30	TFTP	50544	38300	30	Acknowledgement, Block: 1
65	16.217745724	192.168.30.90	192.168.30.80	30	TFTP	38300	50544	30	Data Packet, Block: 2 (last)
72	16.223220780	192.168.40.80	192.168.40.50		TFTP	38033	51673		Data Packet, Block: 10 (last)
75	17.223628281	192.168.30.90	192.168.30.80	30	TFTP	38300	50544	30	Data Packet, Block: 2 (last)
76	19.225399183	192.168.30.90	192.168.30.80	30	TFTP	38300	50544	30	Data Packet, Block: 2 (last)
77	21.228247604	192.168.40.50	192.168.40.80		TFTP	51673	38033		Acknowledgement, Block: 1
78	21.230513658	192.168.30.80	192.168.30.90	30	TFTP	50544	38300	30	Acknowledgement, Block: 1
87	23.230804078	192.168.30.90	192.168.30.80	30	TFTP	38300	50544	30	Data Packet, Block: 2 (last)
88	23.230804298	192.168.30.80	192.168.30.90	30	ICMP	38300	50544	30	Destination unreachable (Port u
89	26.229449088	192.168.40.50	192.168.40.80		TFTP	51673	38033		Acknowledgement, Block: 1
90	26.230826770	192.168.40.80	192.168.40.50		ICMP	51673	38033		Destination unreachable (Port u
91	31.231244411	192.168.40.50	192.168.40.80		TFTP	51673	38033		Acknowledgement, Block: 1
92	31.232189914	192.168.40.80	192.168.40.50		ICMP	51673	38033		Destination unreachable (Port u
93	36.232511835	192.168.40.50	192.168.40.80		TFTP	51673	38033		Acknowledgement, Block: 1
94	36.234168739	192.168.40.80	192.168.40.50		ICMP	51673	38033		Destination unreachable (Port u

Figure 14: Captured traffic during a modified blocknumber of second data packet

Analysis

In this scenario a TFTP client requests a file larger than 512 bytes and manipulates block number of the secound data packet during RRQ traffic. To manipulate the block number for the first data packet containg 1 byte the subsequent data packet is modified. The block number of this data packet is changed to 10 by creating a copy of the original packet and changing the appropiate byte. The manipulated data packet is then forwarded to the TFTP client. The proxy server recivies an acknowledge packet from the TFTP client acknowledging the receipt of the manipulated data packet. The acknowledge packet is forwarded back to the TFTP client.

4. Conclusion and recommendation

The goal of this project was to implement and analyze a VLAN environment using TFTP proxy to manipulate messages exchanged between a client and server. This project has provided valuable insights into traffic manipulation, protocol analysis and network behavior. Through the use of tools like Wireshark, we conclusively verified the effectiveness of our coded manipulation. The intercepted TFTP packets clearly demonstrated our ability to modify network traffic in real-time, confirming the reliability of our approach. Our setup effectively captured the initial RRQ packets from the client and data packets from the server. By executing the faulty situations we can conclude TFTP manages the violations of the protocol, and ensuring data integrity, reliability of data transfer by forcing retrasmissions when incorrect blocknumber sequencing, timout mechanisms to handle delays and error detection capabilities.

Based on our analysis, the outcomes observed in Wireshark confirm that the proxy accurately handles packet manipulation, ensuring what can be describes as seamless communication between the client and the server, despite the intentional modification. By using tools such as Wireshark and Python we enhanced our understanding and gained practical experience in the area of networking.

Recommendations for putting conclusions into practice could be to explore the proxy's capability to handle and manipulate additional protocols beyond TFTP. This could include protocols such as HTTP and other custom protocols, which would involve studying each protocol to understand how to manipulate packets and how it affects network performance.

5. Reference List

Sollins, K.R., 1992. The TFTP protocol (Revision 2). RFC 1350. Internet Engineering Task Force. Available at: https://www.rfc-editor.org/rfc/rfc1350 [Accessed 20 June 2024].