YAŞAR UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING

COMP4920 Senior Design Project II, Spring 2019
Supervisor: Hüseyin Hışıl

METIN: Message Encapsulation over TCP/IP Network

Final Report

(Bachelor of Science Thesis)

15.05.2019

By:
Mert ÖZGEN, Student ID: 16070001190
Mert PEHLİVANCIK, Student ID: 16070001104
Taylan AKKUŞ, Student ID: 15070001054

PLAGIARISM STATEMENT

This report was written by the group members and in our own words, except for quotations from published and unpublished sources which are clearly indicated and acknowledged as such. We are conscious that the incorporation of material from other works or a paraphrase of such material without acknowledgement will be treated as plagiarism according to the University Regulations. The source of any picture, graph, map or other illustration is also indicated, as is the source, published or unpublished, of any material not resulting from our own experimentation, observation or specimen collecting.

Project Group Members:

Name, Last name	Student Number	Signature	Date
Mert ÖZGEN	16070001190		
Mert PEHLİVANCIK	16070001104		
Taylan AKKUŞ	15070001054		

Project Supervisors:

Name, Last name	Department	Signature	Date
Hüseyin Hışıl	Computer Engineering		

ACKNOWLEDGEMENTS

We thank Hüseyin Hışıl, our mentor who supported us in our project since the first day.

KEYWORDS

SSH, firewall, server, client, HTTP request.

ABSTRACT

The emergence of our project is the limitation of some local networks to our SSH server (SSH is a remote management protocol that allows users to control and edit their servers over the Internet.). These restrictions restrict us, and our connection cannot be performed.

If we dig a little deeper, our project is to build an SSH connection to external networks without being caught by the fire wall that is running on the local network. In order to cross the local network bounded by the firewalls, we created a dedicated HTTP-server and assumed that the fire wall allowed only 80 ports to be processed. To do this, we created custom server and custom client configurations that communicate through port 80.

Firstly, our client sends a request to the specially written server via HTTP, where the server sends the information of the SSH server (IP address, username, password) to the server.

Our server, which receives the SSH-Server information, is starting to establish the SSH connection. Once the connection has been established, our server is pushing back to the client screen by returning the data received from the SSH-Server again via the HTTP request.

ÖZET

Projemizin ortaya çıkışı SSH (SSH, kullanıcılara sunucularını internet üzerinden kontrol etmesini ve düzenlemesini sağlayan uzak yönetim protokolüdür) sunucumuza bazı yerel ağların getirdiği sınırlamalardır. Bu sınırlamalar bizi kısıtlıyor ve bağlantımız gerçekleştirilemiyor.

Biraz daha derine inersek, projemizin temeli yerel ağda çalışmakta olan ateş duvarı tarafından yakalanmadan dış ağlara SSH bağlantısı yapmaktır. Ateş duvarının sınırladığı yerel ağı geçebilmek için sadece HTTP konuşan özel bir server oluşturduk ve ateş duvarının sadece 80 portu üzerinden işlem yapmaya izin verdiğini varsayarak ilerledik. Bunu yapabilmek için 80 portundan iletişim kuran özel sunucu ve özel istemci konfigürasyonlarını oluşturduk.

İlk olarak istemcimiz, özel olarak yazılmış sunucuya HTTP üzerinden istek yollayarak sunucu üzerinden bağlanacağımız SSH-Sunucusunun bilgilerini (ip adres, kullanıcı adı, şifre) yolluyor. SSH-Sunucusunun bilgilerini alan sunucumuz, SSH bağlantısını kurmaya başlıyor. Bağlantı kurulduktan sonra, sunucumuz SSH-Sunucusundan aldığı verileri tekrar HTTP isteği üzerinden geri döndürerek istemci ekranına bastırıyor.

TABLE OF CONTENTS

PLAGIARISM STATEMENT ..... **ii**
ACKNOWLEDGEMENTS ..... **iii**
KEYWORDS ..... **iv**
ABSTRACT ..... **v**
ÖZET ..... **vi**
TABLE OF CONTENTS ..... **vii**
LIST OF TABLES ..... **ix**
LIST OF ACRONYMS/ABBREVIATIONS ..... **x**
1. INTRODUCTION **1**
1.1 Description of the Problem 1
1.2 Project Goal 1
1.3 Project Output 1
1.4 Project Activities and Schedule 2
2. DESIGN **2**
2.1 Detailed Design 2
3. IMPLEMENTATION, TESTS and TEST DISCUSSIONS **3**
3.1 Implementation of the Product 3
3.2 Tests and Results of Tests 3
4. CONCLUSIONS **3**
4.1 Summary 3
4.2 Benefits of the Project 3
4.3 Future Work 3
5. References **4**
APPENDICES **5**
APPENDIX A: REQUIREMENTS SPECIFICATION DOCUMENT **5**
1. Introduction **5**
1.1 Purpose 5
1.2 Scope of Project 5
1.3 Glossary 6
2. Requirements **7**
2.1 Establish SSH Access From Private Network To An Outside Computer 7
2.2 User Will Be Send Protocol Commands Through User Interface Application 7
2.3 Configure Port Numbers 7
2.4 Data Will Encrypted 7
2.5 Binary Codes Will Be Encrypted Against Decompilation 7
2.6 Generic Proxy 7
3. Actors and Use Cases **8**
4. Initial Architecture **9**

5. References 10
APPENDIX B: DESIGN SPECIFICATION DOCUMENT 11
1. Introduction 11
2. METIN System High Level Design 12
2.1 METIN System Architecture 12
2.2 METIN System Environments 14
2.3 Simulation Environment 15
3. Detailed System Design 15
3.1 METIN-Client 15
3.2 METIN-WebServer 16
3.3 METIN-SSHAgent 16
4. References 17
APPENDIX C: PRODUCT MANUAL 18
1. Introduction 18
2. Implementation 18
2.1 Source code and Executable Organization 18
2.1.1 METIN-Server 18
2.1.2 METIN-Client 21
2.2 Software Tools 22
2.3 Hardware/Software Platform 22
3. Testing 22
3.1 Result Testing Process 22
3.2 Input Validation Testing Process 23
4. Application Installation, Configuration and Operation 24
5. References 25
APENDIX D: SOURCE CODE/EXECUTABLES/SCRIPTS IN CD/DVD 26

LIST OF FIGURES

Figure 1.4.1 METIN Project Activities 2

LIST OF ACRONYMS/ABBREVIATIONS

RSD	Requirement Specification Document
DSD	Design Specification Document
SSH	Secure Shell
PC	Personal Computer
HTTP	Hyper Text Transfer Protocol

1. INTRODUCTION

1.1 Description of the Problem

Using Secure Shell protocol is blocked in many corporations. However, being able to access remote computer is a requested activity for many reasons. Users can manage their file operations such as reading and writing. Also, they can set their computer settings or start a process remotely.

Today, many organizations provide network security through firewalls. These firewalls block network traffic that can create threats. They generally allow only port 80 to gain capability to their corporations. Therefore, network clients in corporation cannot access to their personal computer where is at their home, so they cannot access their files, configurations and their local processes.

1.2 Project Goal

This project aims to solve described problem (in Section 1.1) which is about network clients who are not able to access their personal computer from corporation's network, remotely.

1.3 Project Output

With this project, it is focused on producing a software application which solve the problem. This software consists of two components. One of them is METIN-HTTP Server and the other is METIN-Client application. The software product was built on the described problem and emerged requirement specification document (RSD) and design specification document (DSD).

In conclusion, we acquired a software with a product manual, an RSD and a DSD.

1.4 Project Activities and Schedule

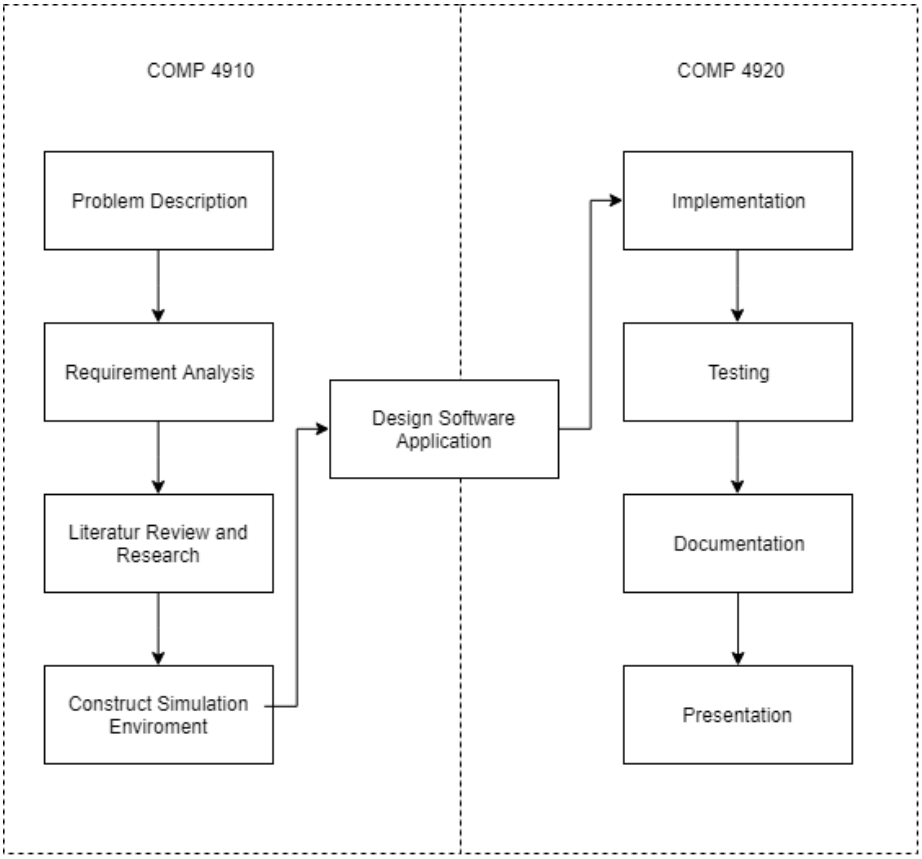

Figure 1.4.1 METIN Project Activities

2. DESIGN

2.1 Detailed Design

To establish connection with tunneling, we considered two running processes. One of them is METIN-Client process which waits input from client and posts this message to METIN-HTTP Server which is second. METIN-HTTP takes over SSH message and dispatches to OpenSSH-Server. The detailed of this design will be illustrated and explained Appendix B, section 5.

3. IMPLEMENTATION, TESTS and TEST DISCUSSIONS

3.1 Implementation of the Product

The detailed of this design will be illustrated and explained in Product Manual in Appendix C in section 2.

3.2 Tests and Results of Tests

The detailed of this design will be illustrated and explained in Product Manual in Appendix C in section 3.

4. CONCLUSIONS

4.1 Summary

We have built our simulation environment on virtual machines running Linux with a configured firewall. Subsequently, we implement the design with C language and was able to connect remote PC which was not blocked SSH traffic. After this test, we try this scenario on our simulation environment. Result was successful. Therefore, it was successful to make SSH traffic over blocked network.

4.2 Benefits of the Project

This project's outcome will provide to users to gain advantage from their work area that is not allowed to use their computer, remotely.

4.3 Future Work

Our ideas about the future, first, we will bring ability to METIN-HTTP Server for providing to listen with multiple client. Thus, multiple users will use their home computer at the same time. Secondly, it is better to interact with application using a better appeared user interface application. Our plan using power of browser. In this way, client will use system without setting up anything. They will only write domain name for the server and will be able to send SSH commands.

5. References

1. M. Ozgen, M. Pehlivancik, T. Akkus, METIN Requirement Specifications Document. Turkey, 2019
2. M. Ozgen, M. Pehlivancik, T. Akkus, METIN Design Specifications Document. Turkey, 2020
3. M. Ozgen, M. Pehlivancik, T. Akkus, METIN Product Manual. Turkey, 2020
4. A. Tanenbaum, Computer Networks, Prentice-Hall, 2011, 5th Ed.

APPENDICES

APPENDIX A: REQUIREMENTS SPECIFICATION DOCUMENT

1. Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of the METIN (Message Encapsulation over TCP/IP Networks). This document is intended for both the stakeholders and the developers of the system.

1.2 Scope of Project

Project name is METIN (Message Encapsulation over TCP/IP Networks). This project aims to create a system that communicates over HTTP and at the mean time encapsulate a hidden TCP/IP protocol in the HTTP traffic without being detected by firewalls. All user interaction will be done through web browser.

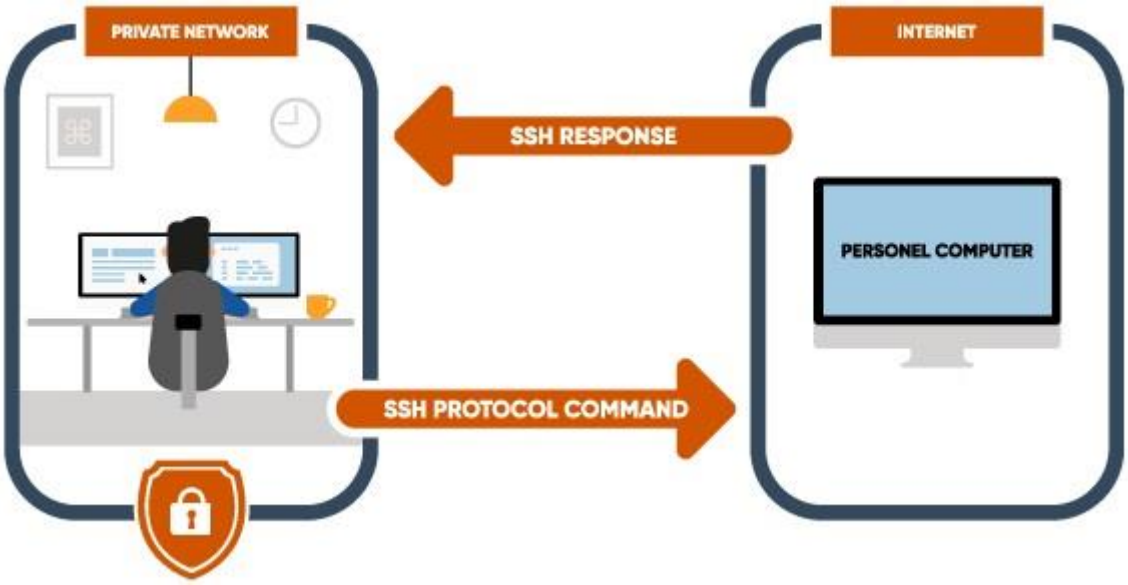

Figure 1.2.1 METIN System Overview

1.3 Glossary

Term	Definition
Data Encapsulation	Data encapsulation refers to sending data where the data is augmented with successive layers of control information before transmission across a network.
Firewall	A firewall is a network security device that monitors incoming and outgoing network traffic and decides whether to allow or block specific traffic based on a defined set of security rules.
Network	A network consists of two or more computers that are linked in order to share resources, exchange files, or allow electronic communications.
Network Protocols	Network protocols incorporate all the processes, requirements and constraints of initiating and accomplishing communication between computers, servers, routers and other network-enabled devices.
Software Requirements Specification	A document that completely describes all the functions of a proposed system and the constraints under which it must operate. For example, this document.
Proxy Server	A proxy server is a dedicated computer or a software system running on a computer that acts as an intermediary between an endpoint device, such as a computer, and another server from which a user or client is requesting a service.
Virtualization	Virtualization refers to the creation of a virtual resource such as a server, desktop, operating system, file, storage or network.

Table 1.3 Glossary

2. Requirements

2.1 Establish SSH Access From Private Network To An Outside Computer

A computer, which is placed in a private network, will establish ssh connection with by-passing network's firewall. The system will connect itself to remote pc due to account’s IP address and port number.

2.2 User Will Be Send Protocol Commands Through User Interface Application

User can send protocol commands user interface. This interface will be a console on desktop.

2.3 Configure Port Numbers

User can configure the port number. It means user will be give an input as a port number on interface.

2.4 Data Will Encrypted

The data, which will be sent, must be encrypted.

2.5 Binary Codes Will Be Encrypted Against Decompilation

There will be private key of encryption so, there must be encrypted binary code to prevent reverse engineering, because after decompilation the private key will be explicit.

2.6 Generic Proxy

You can configure a generic proxy for HTTP/HTTPs if it is configured to use the SOCKS protocol.

3. Actors and Use Cases

Actors	Description
User	The user will use system to access remote home computer that is out of campus network.

Table 3.1 Actors

Use Case	Description
Establish SSH Connection	User will be able to establish a SSH connection with a remote machine that is outside or inside of a network that is protected by firewall. and all ssh protocol commands will be sent correctly.
Send Protocol Commands	After comminication channel established, user will send protocol commands.
Configure Port	User will be able to change port number that will be connect.

Table 3.2 Use case

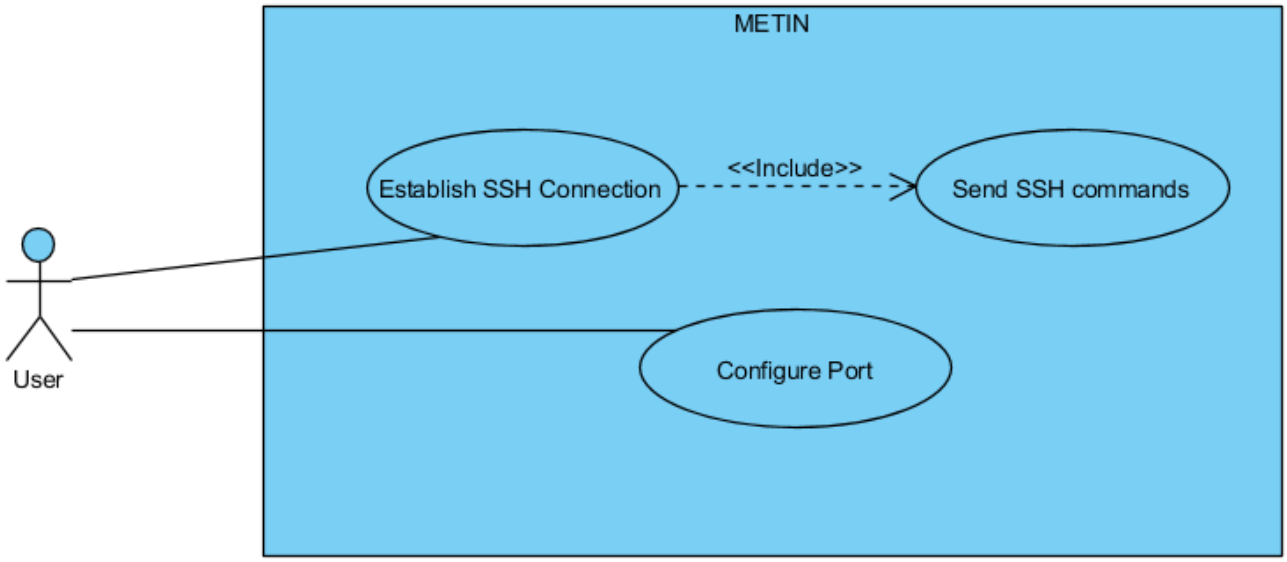

Figure 3.1 METIN Use Case Diagram

4. Initial Architecture

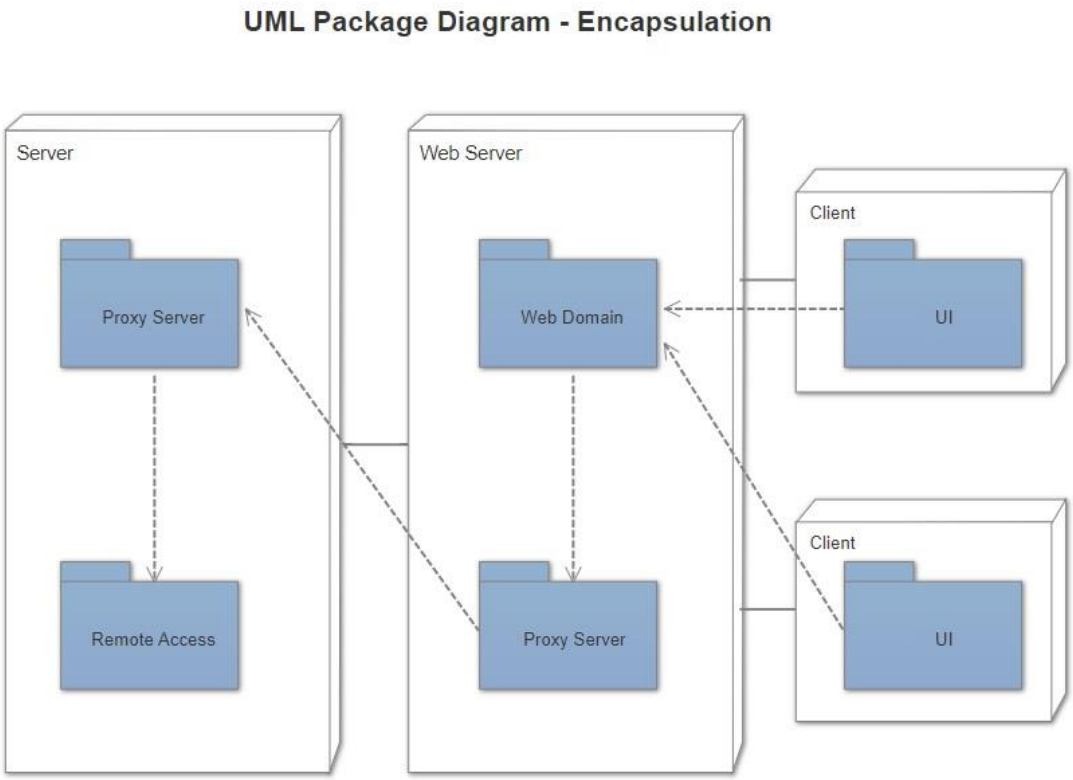

Figure 4.1 Package Diagram

5. References

1. A. Tanenbaum, Computer Networks, Prentice-Hall, 2011, 5th Ed.

2. W. Stallings, Data and Computer Communications, Pearson, 2013, 10th Ed.

3. C. Cowan, P. Wagle, C. Pu, S. Beattie, and J. Walpole, “Buffer overflows: Attacks and defenses for the vulnerability of the decade,” in DARPA Information Survivability Conference and Exposition, 2000. DISCEX’00. Proceedings, vol. 2. IEEE, 2000, pp. 119–129.

APPENDIX B: DESIGN SPECIFICATION DOCUMENT

1. Introduction

The purpose of the software project is to develop a console application that will be able to communicate HTTP in C and Java and LINUX environment.

1. User Interface
2. Network Communication
3. Encryption

The design is based on METIN Requirements Specification Document, Revision 3.2, in file METIN v3.1[1]. The system architecture and overall high-level structure of METIN console application are given in Section 4. Design details of all application functions and the user interface in terms of are methods of all classes will later be given in Section 3 of this document.

2. METIN System High Level Design

2.1 METIN System Architecture

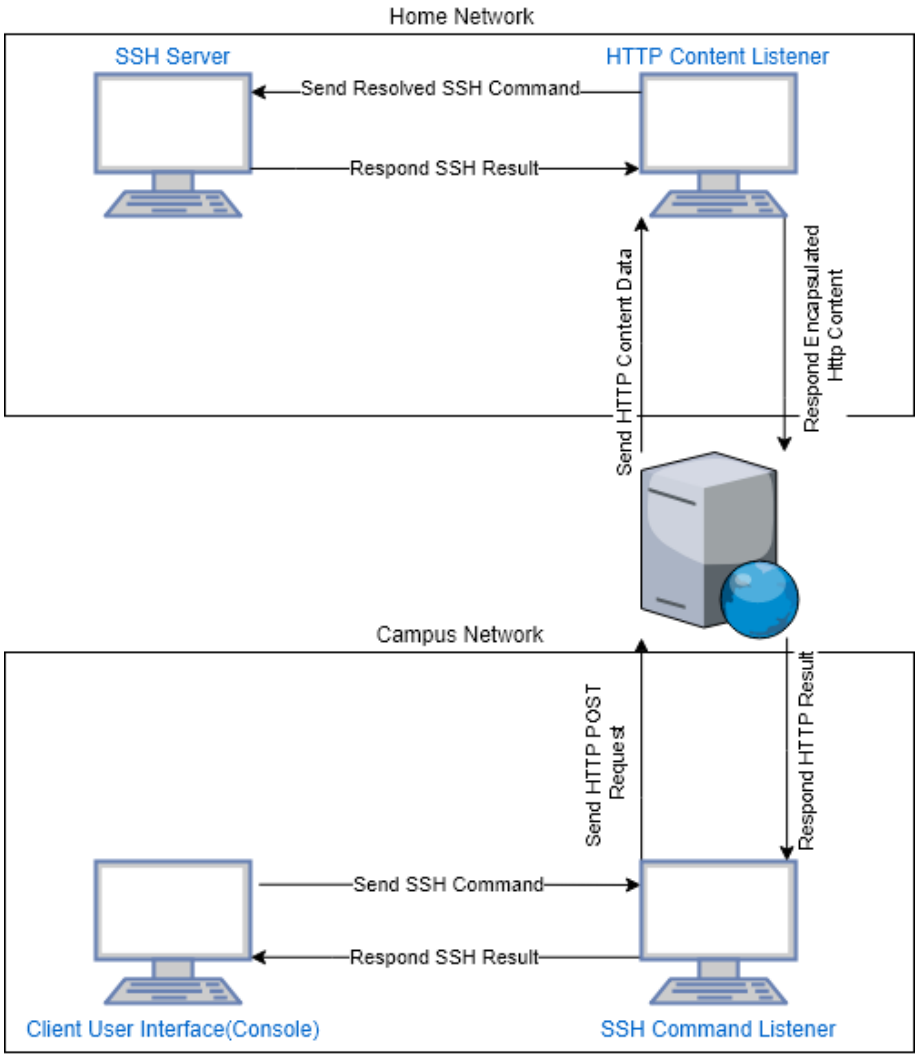

Figure 2.1.1 System Architecture

Figure 2.1.1 shows system architecture that will explain how system components will act with each other. First, client at campus network will send an SSH command on user interface process and this program will send this message “SSH Command Listener” process that is always listening an SSH command. Then, this message will be encapsulated as HTTP request on this step. Then, request will be sent to Web Server. While “SSH Command Listener” process wait a response, web server will send HTTP content to “Web Server Listener” process. This process takes content of data and decapsulates it then it will send “SSH Server” process. This process will execute command and will return a response to back. After that, each process will respond back the SSH result step by step until response will reach to client.

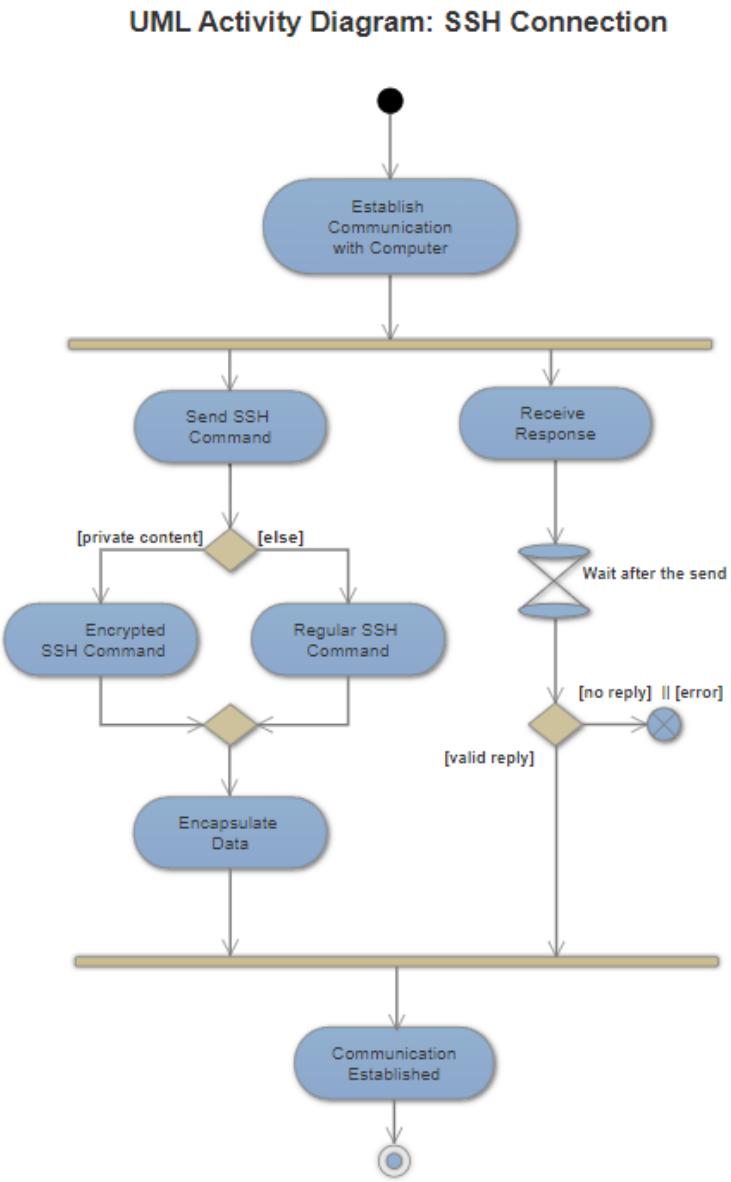

Figure 2.1.2 SSH Connection Activity Diagram

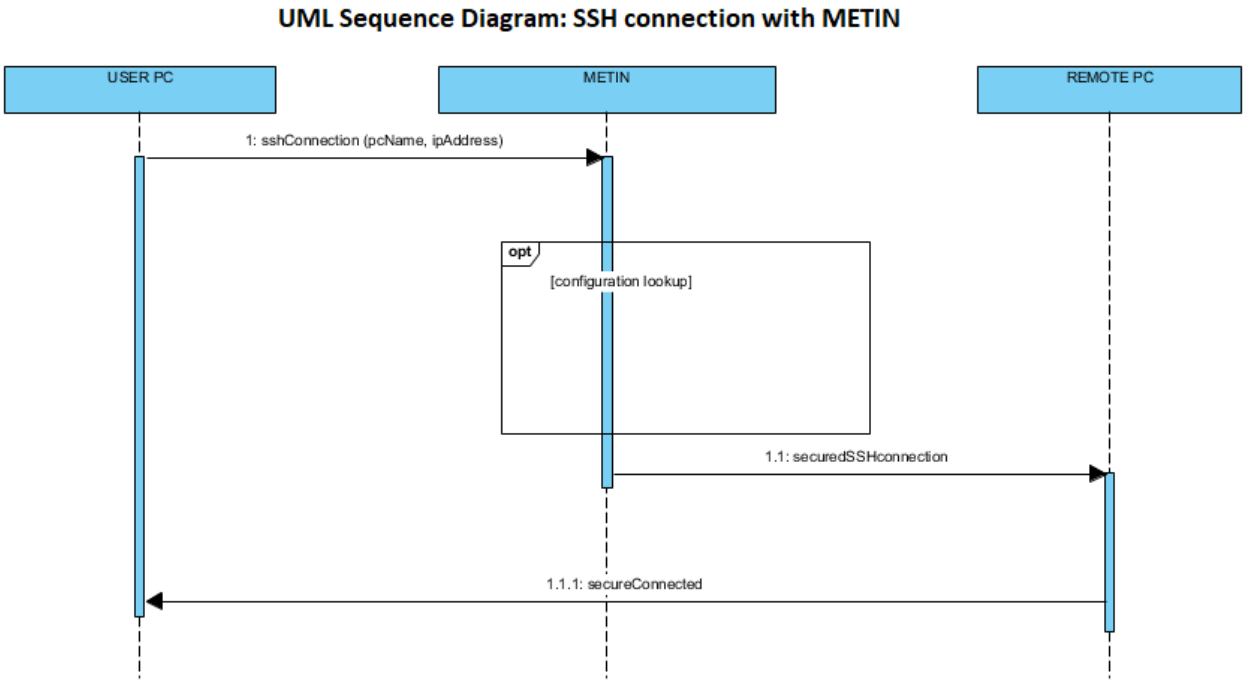

Figure 2.11.3 SSH Connection Sequence Diagram

2.2 METIN System Environments

The most important material is the computer,which use generally by team member of project. The jobs that doing by project members need running on a computer that has high processor power and high capacity memory. In order to developed networking jobs, making virtualization is needed. Therefore, it is need to have at least 4 gb memory and i3 processor in this computers. On the other hand, there is no need to have high video processing power so the cost of computer systems will be cheaper. The decision of the project members are using Windows and Debian based operating systems to use on computers. These operating systems will be Windows 10 and Ubuntu 16.04.5. In addition, for the computers which are with low system performance, it will use Ubuntu Server 16.04.5. The team members decided that it should have a system developed in C or Java programming language. One of the reasons for the choice of C or java programming language was that it is portable across different hardware platforms and the company wants software that could run on PCs.

When the team study on computer networking, there will be need to simulation of network. The best way of overviewing of network traffic is using Cisco Packet Tracer. It is a tool that simulate each data packets and behaviour of each network device. Moreover, to make more realistic simulation, it will be used virtualization technology which is "Type II" Hypervisor. The hypervisor will be Oracle Virtualbox.

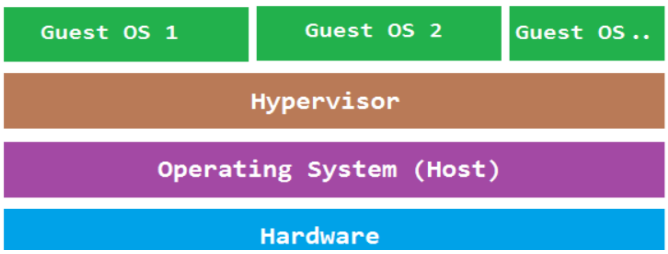

Figure 2.2.1 Type 2 Hypervisor

2.3 Simulation Environment

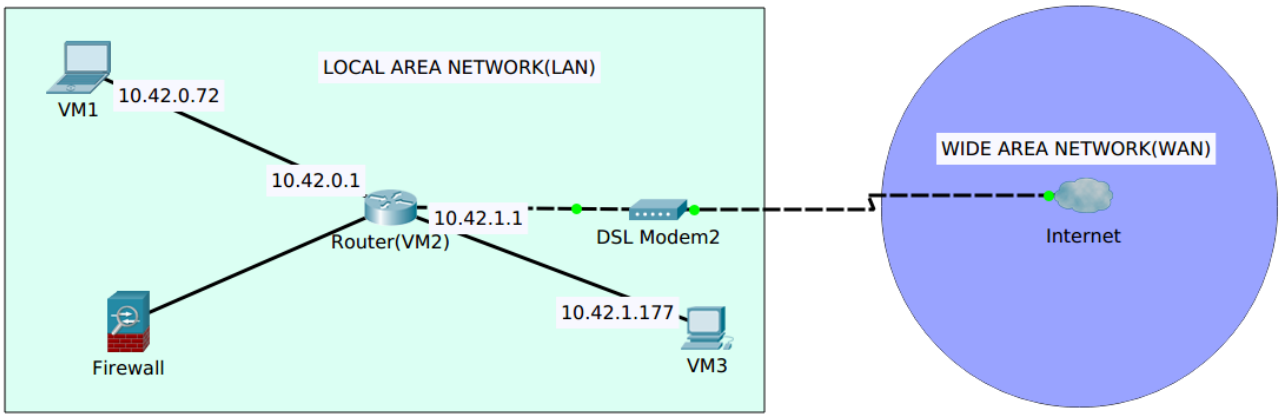

Figure 2.3.1 Network Diagram

Figure 2.3.1 shows network diagram of simulation environment that is necessary to test application. This simulation is similar with real campus network. All transitions are transferred over router which runs a firewall. With taking reference this diagram we are running this simulation using Virtual Machines. We can set up rules on firewall and try to bypass these rules.

3. Detailed System Design

The system design consists of three processes that are METIN-Client, METIN-Web Server and METIN-SSH Agent. All of three process will be explained in following sections.

3.1 METIN-Client

The application will get input from user. The message will be encapsulated then it will be sent to METIN-Web Server and will wait for response. After the response is received, it will be shown as an output to user and again wait for input.

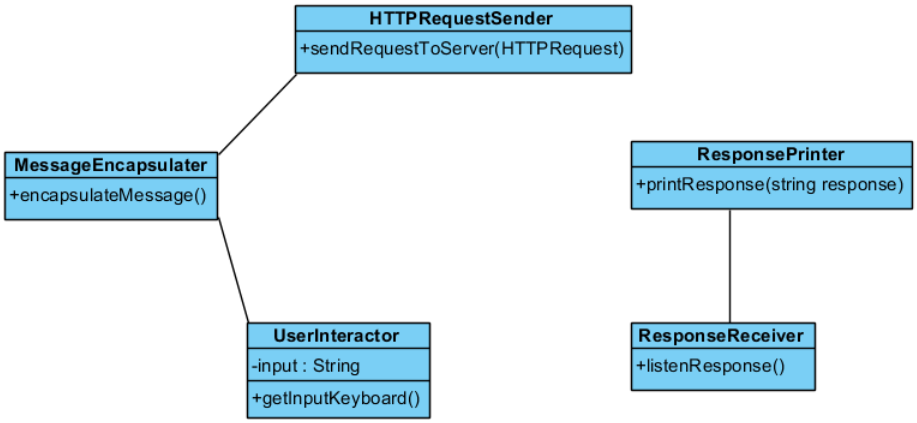

Figure 3.1.1 METIN-Client Class Diagram

3.2 METIN-Web Server

METIN-Web Server listens HTTP request. After getting a request, body will be parsed from it. Parsed body will send to METIN-SSH Agent process as SSH command over a TCP socket then wait for reading result that will come from METIN-SSH Agent. Again, the result will encapsulate as a HTTP response to reply the request. Figure 5.2.1 shows class diagram of METIN-Web Server.

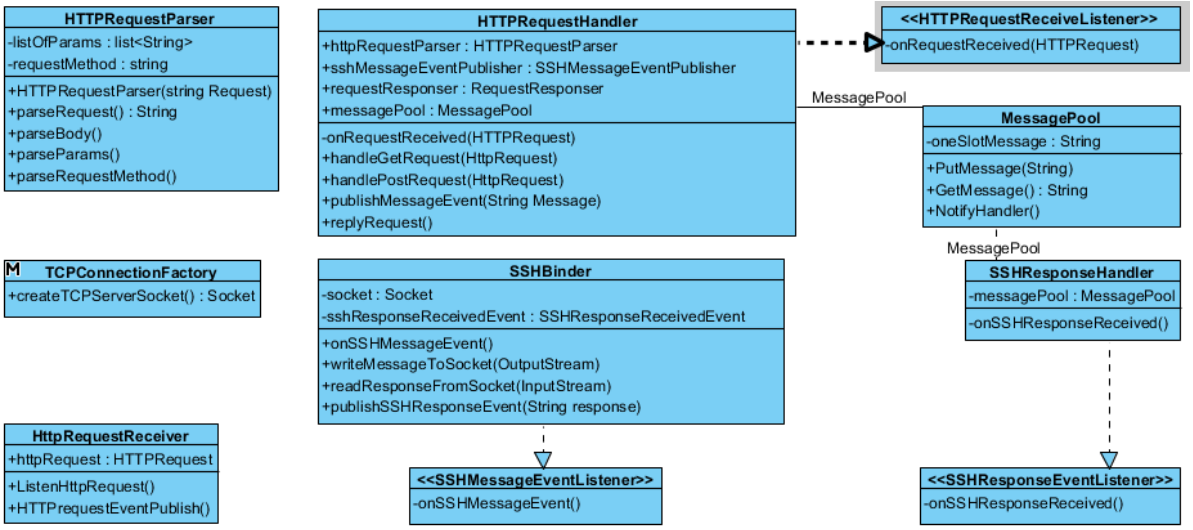

Figure 3.2.1 METIN-Web Server Class Diagram

3.3 METIN-SSH Agent

This process is a middle-ware that can open SSH session. The received message that come from server will be dispatched through channel that can be accepted by OPENSSH-Server. This process uses 3rd part library that can do SSH traffic. The message will be sent to OPENSSH-Server and it return result then process send back this result without any change.

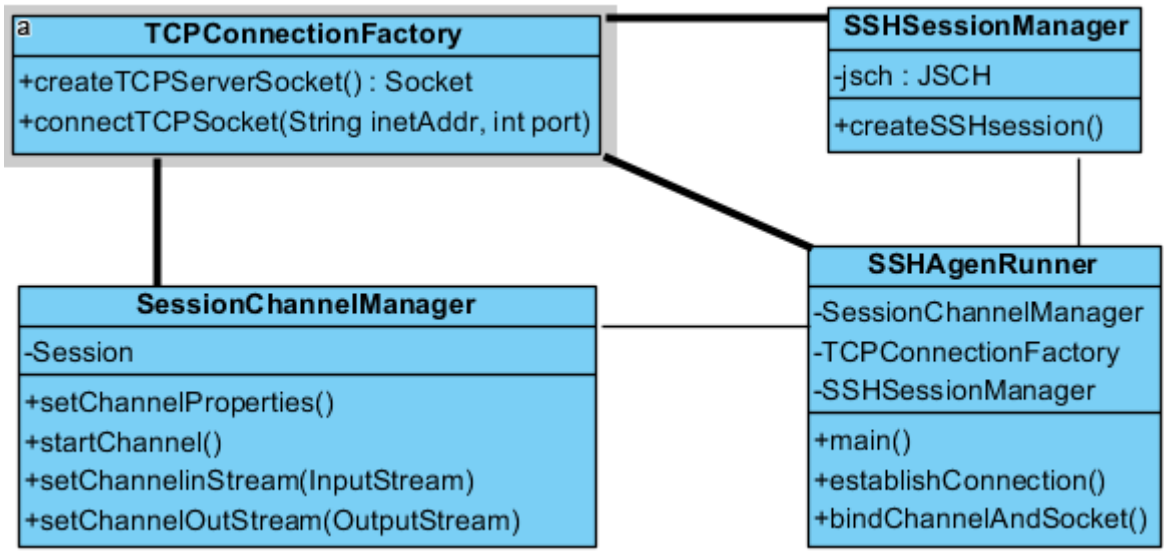

Figure 3.3.1 METIN-SSH-Agent Class Diagram

4. References

1. M. Ozgen, M. Pehlivancik, T. Akkus, *METIN Requirement Specification Document*. Turkey, 2019.
2. A. Tanenbaum, Computer Networks, Prentice-Hall, 2011, 5th Ed.
3. W. Stallings, Data and Computer Communications, Pearson, 2013, 10th Ed.

APPENDIX C: PRODUCT MANUAL

1. Introduction

The purpose of this product manual is to document the implementation, testing, installation and operation of METIN: Message Encapsulation over TCP/IP Networks as a software product.

METIN Application is implemented and tested as it is described in METIN Application - Design Specification Document, Revision 3.0, satisfying the requirements in METIN Application - Requirements Specification Document, Revision 2.0.

Implementation, testing and operation details are given in the following sections of this document.

2. Implementation

2.1 Source code and Executable Organization

2.1.1 METIN-Server

The METIN-Server was developed with the C programming language. There are SSH-Client available within HTTP Server in METIN-Server. The libssh library is used for the SSH client. The HTTP Server and the SSH Client are connected and run simultaneously with each other. Otherwise, a certain part of the stream is encrypted with AES-256 to provide better security. Now, important parts of the source code will be explained step by step.

For our example blocks, codes are presented in here, and then are commented on key parts of the codes. So, here is the first code block, saved in the file “libssh.c”.

```
1.  #include <stdlib.h>
2.  #include <stdio.h>
3.  #include <libssh/libssh.h>
4.  #include <unistd.h>
5.  #include <sys/types.h>
6.  #include <sys/socket.h>
7.  #include <netinet/in.h>
8.  #include <netdb.h>
9.  #include <arpa/inet.h>
10. #include <string.h>
11. #include <sys/stat.h>
```

```
12. #include <fcntl.h>
13. #include "free_channel.h"
14. #include "free_session.h"
15. #include "error.h"
16. #include "shell_session.h"
17. #include "new_session.h"
18. #include "execute_command.h"
19. #include "interactive_shell_session.h"
20. #include "setHTTPHeader.h"
21. #include "report.h"
22. #include "server.h"
23. #include "ConnectServerAndSSHSession.h"
24. #include "recursive.h"
25. #define SIZE 1024
26.
27. int main(int argc, char** argv) {
28.
29.             int server_port = 0;
30.             char IP[15];
31.             memset(IP, 0, 15);
32.             printf("Server IP Address: ");
33.             scanf("%[^\n]%*c", &IP);
34.             printf("Server Port: ");
35.             scanf("%d", &server_port);
36.             recursive(server_port, IP);
37.              return 0;
38.  }
```

The block of code above is the main block of code and the program starts here. Required libraries and used header file is initialized. In the Main block, two variables, which are “server_port” and “IP”, are defined and initialized. If the initialization is not done, the variable contains any value from the memory and may prevent the program from functioning properly. In here, the IP address and port number, which are used by HTTP Server, are got from user and these are assigned to the “recursive()” function as a parameter.

Below is the second code block, saved in the file “recursive.h”.

```
1.  #ifndef RECURSIVE_H_
2.  #define RECURSIVE_H_
3.  #include "shell_session.h"
4.  int recursive(int server_port, char* IP) {
5.          server(server_port, IP);
6.          ConnectServerAndSSHSession();
7.          int rv = 0;
8.          rv = new_session(username, password, hostname, server_port, IP);
9.
10.         if (shell_session(session, server_port, IP) != SSH_OK) {
11.                 error(session, server_port, IP);
12.                 free_channel(channel);
13.                 return rc;
14.         }
15.         interactive_shell_session(session, channel, server_port, IP);
16. }
17.
18.
19. #endif /* RECURSIVE_H_ */
```

This header file contains a function, which name is recursive, that returns an integer and takes two arguments. This function is the basic structure of the program and it includes many functions. First called function name is “server” and this function takes two arguments, which are “server_port” and “IP” for starting HTTP server. The IP address and server port can be changed by the user and this allows the user to run server at the IP address and port requested. The operating principle of the server is shown in the following state diagram:

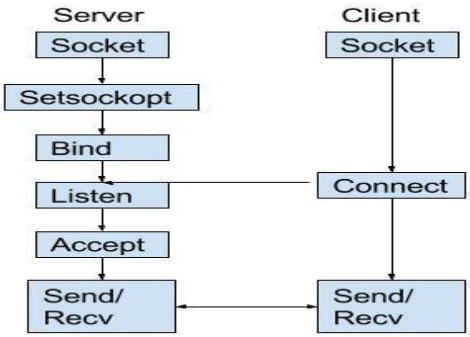

Figure 11.1.1 State Diagram for Server and Client

These steps are repeated exactly. Firstly, TCP socket is created and bind the socket to server address using bind(). Server socket is put in a passive mode, where it waits for the client to approach the server to make a connection using “listen()”. At this point, connection is established between client and server, and they are ready to transfer data using “accept()”.

Returns the recursive function after the “server()” and “ConnectServerAndSSHSession()” is called in “recursive()” function. In here, the encrypted message, which comes from the client, is read and message is decrypted. Then the decrypted message is decoded and assigned to three different char arrays: username, password and hostname. These char arrays were defined as global variable and these will be used by SSH Client to connect the SSH Server.

After the “ConnectServerAndSSHSession()” finishes, the recursive function is resumed and “new_session()” is called in “recursive()” function. The most important object in an SSH connection is the SSH session. The ssh_options_set() function sets the options of the session. The most important options are:

- SSH_OPTIONS_HOST: the name of the host you want to connect to
- SSH_OPTIONS_PORT: the used port (default is port 22)
- SSH_OPTIONS_USER: the system user under which you want to connect

While give parameters to ssh_options_set(); char arrays, which are username, password and hostname, is used. Lastly, the client will be authenticated with ssh_userauth_password(). The classical ways are password for this. While “new_session()” is ended, “shell_session()” is called and channel is created for SSH in here.

Function, which called lastly in the recursive(), is “interactive_shell_session()”. Data is exchanged with SSH-Server via pre-built channel with this function. Data transferred from the SSH-Server to the SSH-Client is sent to the HTTP-Client via HTTP-Server. While the user sends the command to SSH-Server, user send command to HTTP-Server via HTTP-Client and command is transmitted to SSH-Client by HTTP-Server. Later, command is run by SSH-Client and the output of the command is sent by the same steps. In the event of a possible error, the “error()” function is called. This function is important for continuity of HTTP Server. In here, all of connection is closed and “recursive()” function is called once again.

2.1.2 METIN-Client

The METIN-Client is the customized version of the HTTP Client. Firstly, METIN-Client establishes a connection with HTTP-Server in METIN-Server. After the establishing connection, the commands are sent to the METIN-Server to run on the SSH-Server. The outputs of these commands from METIN-Server is taken by METIN-Client and these outputs is printed to program by METIN-Client. To establish a connection with the HTTP-Server in METIN-Server, the IP address of the server and the port number, which is listened by the server, to are requested from the user. The values received from the user are transmitted to the “ConnectHTTPServer()” function in “HTTPClient.c” and HTTP-Server in METIN-Server is established a connection by this function. The function “CreateSSHSession ()“ in “HTTPClient.c“ is called and the username, password and SSH-Server IP address required for the METIN-Server to connect to the SSH-Server, so these values is taken, and these are encrypted with AES-256 for security purposes. When the encrypted data is sent to the METIN-Server with “SendHTTPRequest()” function, data, which is received by METIN-Server, is received by “ReadResponse()”.

2.2 Software Tools

- C compiler.
- Libssh library in C.

2.3 Hardware/Software Platform

The most important material is the computer, which use generally by team member of project. Therefore, it is needed to have at least 4 GB memory and i3 processor in these computers.

The decision of the project members is using Windows and Debian based operating systems to use on computers. These operating systems will be Windows 10 and Ubuntu 16.04.5. In addition, for the computers which are with low system performance, it will use Ubuntu Server 16.04.5.

3. Testing

We proposed two test processes. One of them is testing equality of the SSH result for same file using three type of SSH connection. Second is testing input validation process of METIN-Client for correct output according to input states.

3.1 Result Testing Process

The figure shows test process that is for checking output results of SSH commands. To explain, an SSH message (ls -l) is sent using three system environments for same directory and same file. Test result is valid when three results are same.

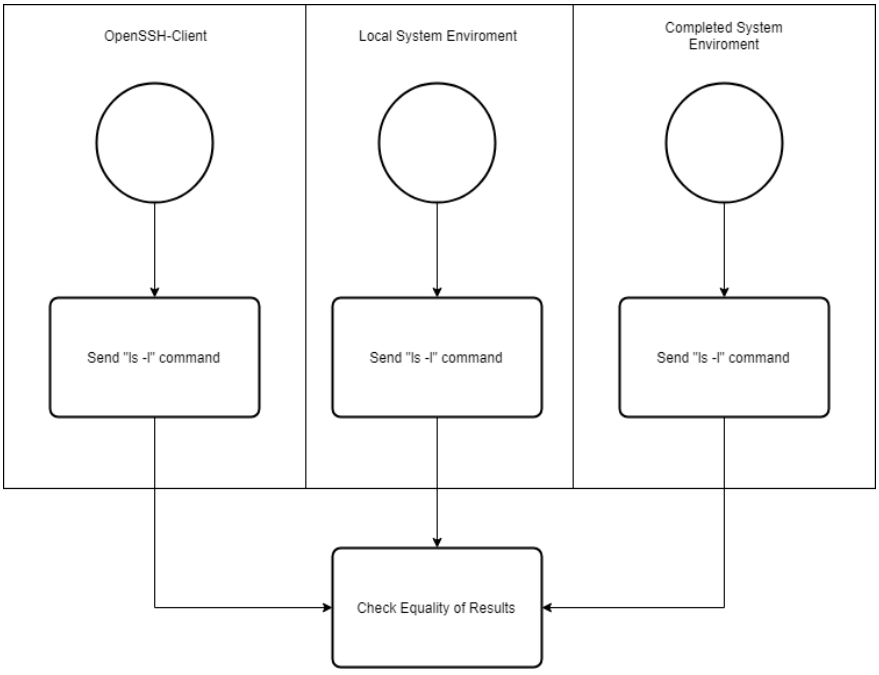

Figure 3.1.1 Result Testing Process Flowchart

3.2 Input Validation Testing Process

Figure shows input validation process. METIN-Client requests from client to input three information; hostname, username and password. After input provided from user, the error messages must be shown when an invalid input. Table shows which error message will be shown according to type of invalid input.

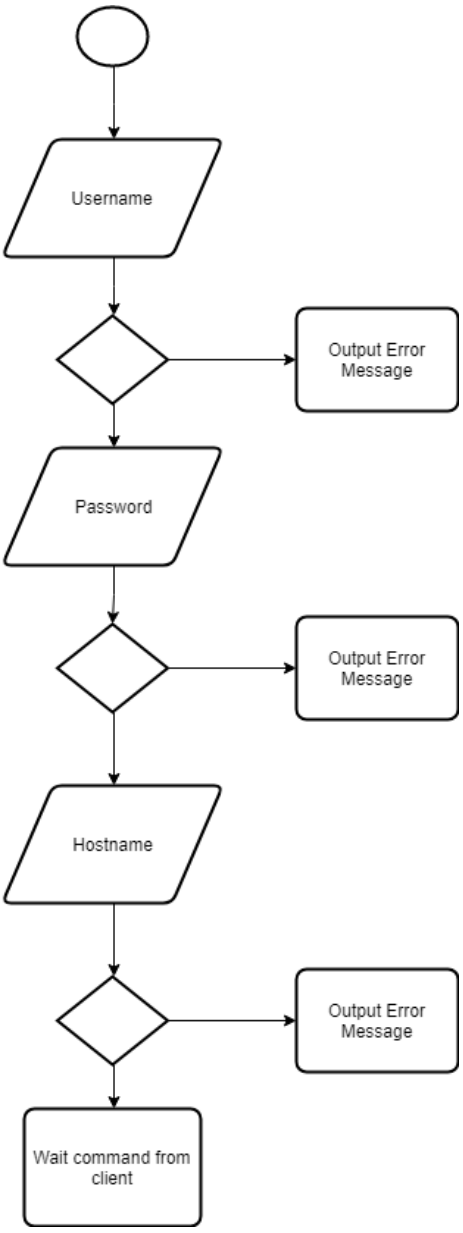

Figure 3.2.1 Input Validation Testing Process Flowchart

#	Username	Password	Hostname	Expected Error Message
1	Correct	Correct	Incorrect	Access denied. Authentication that can continue public key, password
2	Correct	Incorrect	Correct	Access denied. Authentication that can continue public key, password
3	Incorrect	Correct	Correct	Error: failed to resolve hostname {hostname}. (Name or service unknown)
4	Correct	Correct	Null	Hostname required

Table 3.2.1 Expected Values

4. Application Installation, Configuration and Operation

Ubuntu:

- You must install “libssh” library before you start configuration. You can follow the installation here. (www.libssh.org/get-it/)
- You need port forwarding because your computer as a server for other devices.
- The 80 port of the modem must be switched on.
- You must download the METIN source code from GitHub.(https://github.com/metin35)
- After downloading the source code, you compile and run the "SERVER" file on the external network.
- The server started listening to HTTP Requests.
- If the server settings are done, configure the computer settings on the internal network.
- You compile and run the “CLIENT” file on the internal network client.
- When running the client, it will need the server’s IP address and port.

  (./client 54.232.12.45 80)
- When the client runs, it will prompt for the Username, Password and , IP Address of the computer that you want to connect to.
- Connection will be provided.

5. References

1. M. Ozgen, M. Pehlivancik, T. Akkus, METIN Requirement Specifications Document. Turkey, 2019
2. M. Ozgen, M. Pehlivancik, T. Akkus, METIN Design Specifications Document. Turkey, 2020
3. A. Tanenbaum, Computer Networks, Prentice-Hall, 2011, 5th Ed.

APENDIX D: SOURCE CODE/EXECUTABLES/SCRIPTS IN CD/DVD