Bachelor Thesis

DESIGN AND IMPLEMENTATION OF CCEAP USER INTERFACE

Bachelor Thesis Submitted as Partial Fulfillment of the Requirements for the Degree of B.Sc. in Applied computer science at the University of applied sciences (Hochschule Worms)

by

Feugang Kemegni Fabrice

Submitted Summer, 2017
supervised by Prof. Dr. Steffen Wendzel

Applied Computer Science

Software construction.

Declaration of Authenticity

I hereby declare that all material presented in this document is my own work. I fully and specifically acknowledge wherever adapted from other sources.

I understand that if at any time it is shown or proven that I have intentionally misrepresented material present here, any decree or credits awarded to me on the basis of the material may be revoked.

I declare that all statements and information contained here is true, correct and accurate to the best of my knowledge and belief.

Ludwigshafen, Summer 2017.

…………………………..

Feugang Kemegni Fabrice

Abstract

This Bachelor thesis discusses the development of a graphical user interface for a command line program CCEAP.

CCEAP (Covert Channel Educational Analysis Protocol) is a networking protocol in the form of a command line program to help learn/teach covert channels and hiding patterns.

The cceap program is currently executed from the command line, by starting a cceap-server with a given port number of a computer, then on another command line starting the cceap-client program with the IP address and port number of the computer where the server is running. The client is started with some other parameters that correspond to the cceap's protocol settings.

The first Chapter of this thesis is a discussion about what cceap is, and the related concepts connected to it. Additional background information required to understand the purpose of cceap are discussed in this here. Some limitations of the cceap leading to the need for a complementary graphical user interface are also presented.

The second chapter discusses the implementation phase of the Graphical User interface tool, including design decisions as well as some of the important portions of the code. This section is followed a chapter on that discussion of some problems/issues faced during the implementation of the program as well as the solutions to some of the problems.

In the last section, extensions to the GUI tool are presented. These refer to the features that could be added to the program during future development potentially by other students that might be involved in the evolution of CCEAP. This is then followed by my conclusion, a bibliography and a list of figures.

Contents

1 Introduction

1.1 What is CCEAP

To simplify the understanding of the term **CCEAP**, it is indispensable to firstly define some of the concepts that are related to it. These concepts include covert channels, hiding patterns, network steganography.

1.1.1 Steganography/Covert Channels and Hiding patterns

Steganography is the practice of concealing a file, message, image, or video within another file, message, image, or video. The word steganography combines the Greek words **steganos** meaning "covered, concealed, or protected", and **graphein** meaning "writing". In other words, it is the art of hiding the fact that communication is taking place, by hiding information in other information.

Covert Channels also called network steganography hiding methods, are techniques used to conceal communications within network traffics. Hiding communications in plain sight by using public media (in this case the network traffic) to practice secret communication. Secret communication in the sense that the fact that the communication is even taking place is unknown. There are more than a hundred of such network steganography methods. however as shown in (Wendzel et al., 2015) these can be summarized into 11 so called **Hiding Patterns**.

1.1.2 CCEAP

With such a brief understanding of what a covert channel in the domain of networking means and without deeply diving into the subject, we can now jump to what CCEAP stands for and why are its purposes.

CCEAP (**Covert Channel Educational Analysis Protocol**) is a command line tool(program) that emulates an application layer protocol and is designed to help in teaching/learning covert channels. It is an application layer protocol that uses Ethernet packets payloads as covert channel (the term covert channel is explained above).

The idea behind CCEAP is to provide such a protocol that will allow learners to model/manipulate this protocol's structure with the aim of creating a covert channel.

The main features of cceap is that it is based on hiding patterns (Wendzel et al., 2015), which is summarizes more than 120 network steganographic methods to only 11 hiding patterns.

Thus, providing an economic approach to learn only these 11 hiding patterns which summarize the hundred methods.

Additionally, cceap combines all methods of hiding data in network communications into a single protocol instead of using each separate protocol i.e. IP, UDP, HTTP, TCP etc. to demonstrate how covert channels work for each of them. [1]

1.2 How does CCEAP function

Now that the term CCEAP has been explained, a discussion about how it functions is in order. The CCEAP is a cross platform command line tool that runs on windows, Unix Linux and MAC OS. Detailed explanations on the installation steps and other Instructions on running the CCEAP tool together with some practical exercises to demonstrate the functioning of the tool can be found on this webpage CCEAP documentation
ttps://github.com/cdpxe/CCEAP/tree/master/documentation.
However here is a shorthand on how to run the tool on the most widely used operating systems is provided below, but it is highly recommended to visit the CCEAP's website.

After downloading cceap from the CCEAP's website, https://github.com/cdpxe/CCEAP.git into a folder on the computer, follow the steps below depending on the architecture.

1.2.1 Running CCEAP on windows

- open a command line tool **cmd**

- navigate to the folder where the CCEAP tools are stored i.e. cd /path/to/CCEP/Folder/

- start server.exe -P 1234

- start client.exe -P 1234 -D 127.0.0.1

1.2.2 Running CCEAP on Unix

- open shell (Terminal)

- with the terminal, navigate to folder where CCEAP program is stored

- start server
 ./server -P 1234

- start client
 ./client -p 1234 -D 127.0.0.1

---

[1]

In the example above, **-P** is the Port number, which must be the same for client and server,

-**D** is the IP address of the computer where the server is running, if you are running the client and server on different computers, you need to replace the 127.0.0.1 with the actual address of the computer where the server is running. Visit the CCEAP website https://github.com/cdpxe/CCEAP/tree/master/documentation for more details about the parameters.

The figures below show cceap client and server running on a Debian machine. The output is similar on other Unix computers and windows. Figure 1 shows the cceap-server started with parameter -h, this displays the help menu and nothing else. Figure 2 similarly starts the client with parameter -h, which also simply displays the help menu to user.
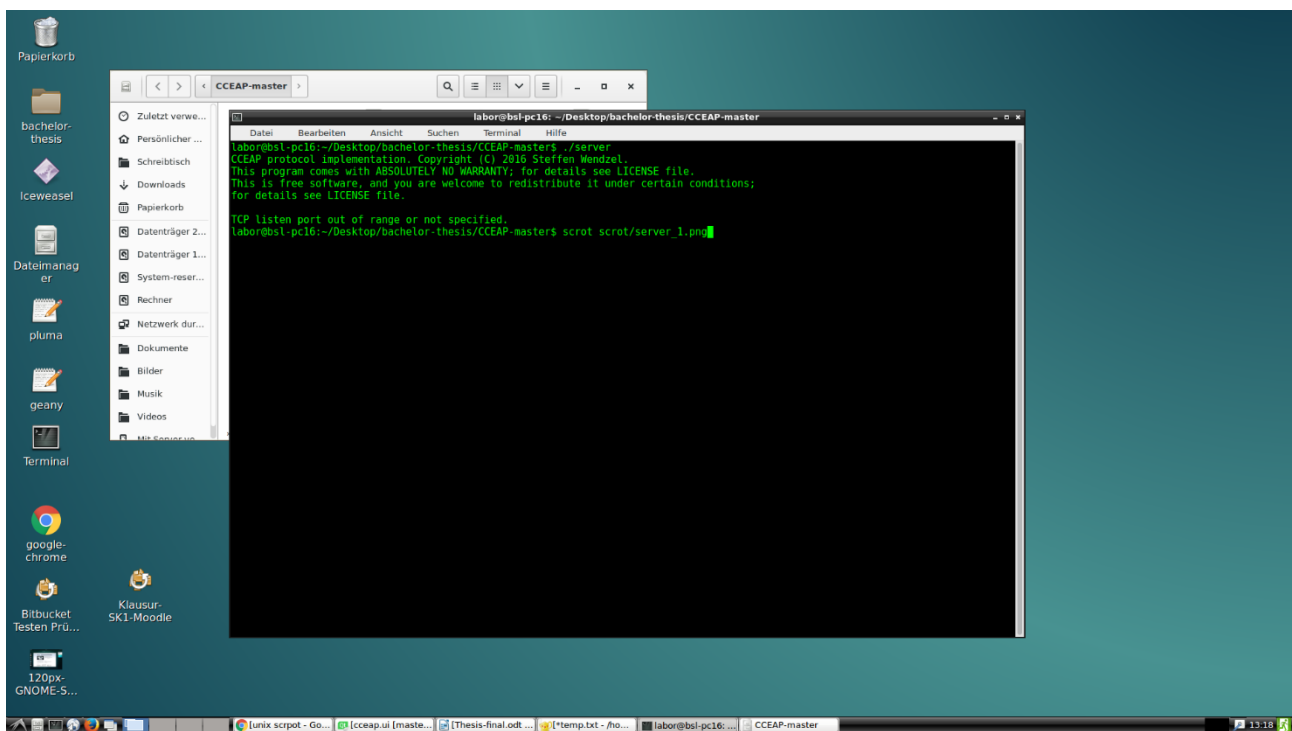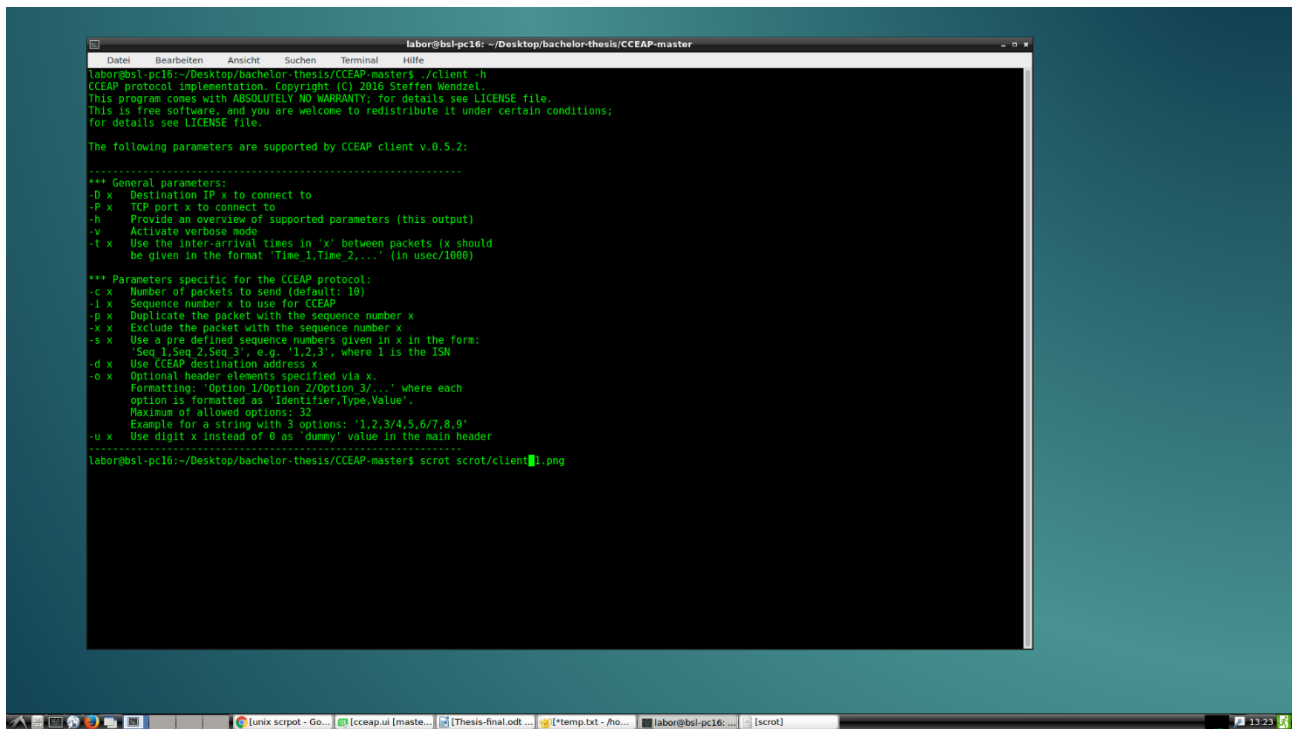

Figure 1: server -h

Figure 2 below demonstrates the cceap-client's output when started with parameter -h which simply displays the option menu.

```
labor@bsl-pc16:~/Desktop/bachelor-thesis/CCEAP-master$ ./client -h
CCEAP protocol implementation. Copyright (C) 2016 Steffen Wendzel.
This program comes with ABSOLUTELY NO WARRANTY; for details see LICENSE file.
This is free software, and you are welcome to redistribute it under certain conditions;
for details see LICENSE file.

The following parameters are supported by CCEAP client v.0.5.2:

----------------------------------------------------------
*** General parameters:
-D x   Destination IP x to connect to
-P x   TCP port x to connect to
-h     Provide an overview of supported parameters (this output)
-v     Activate verbose mode
-t x   Use the inter-arrival times in 'x' between packets (x should
       be given in the format 'Time_1,Time_2,...' (in usec/1000)

*** Parameters specific for the CCEAP protocol:
-c x   Number of packets to send (default: 10)
-i x   Sequence number x to use for CCEAP
-p x   Duplicate the packet with the sequence number x
-x x   Exclude the packet with the sequence number x
-s x   Use a pre defined sequence numbers given in x in the form:
       'Seq_1,Seq_2,Seq_3', e.g. '1,2,3', where 1 is the ISN
-d x   Use CCEAP destination address x
-o x   Optional header elements specified via x.
       Formatting: 'Option 1/Option 2/Option 3/...' where each
       option is formatted as 'Identifier,Type,Value'.
       Maximum of allowed options: 32
       Example for a string with 3 options: '1,2,3/4,5,6/7,8,9'
-u x   Use digit x instead of 0 as 'dummy' value in the main header
----------------------------------------------------------
labor@bsl-pc16:~/Desktop/bachelor-thesis/CCEAP-master$ scrot scrot/client1.png
```
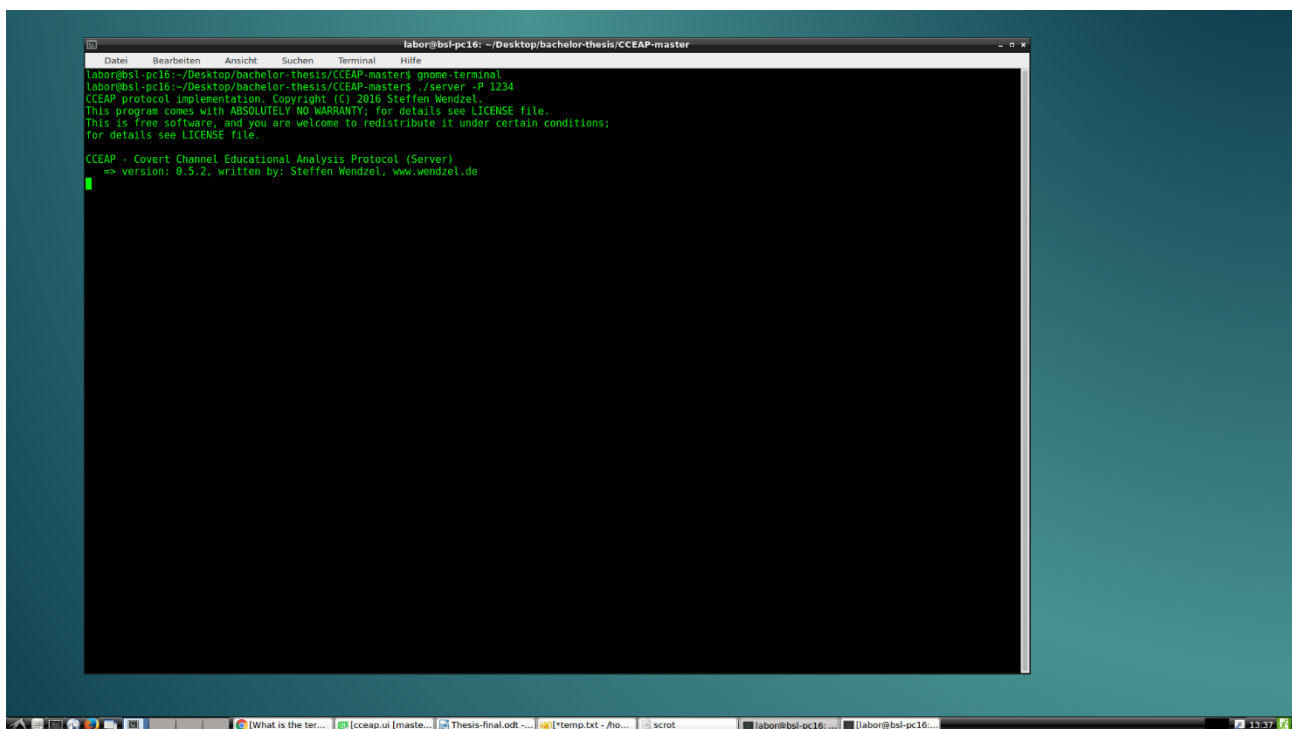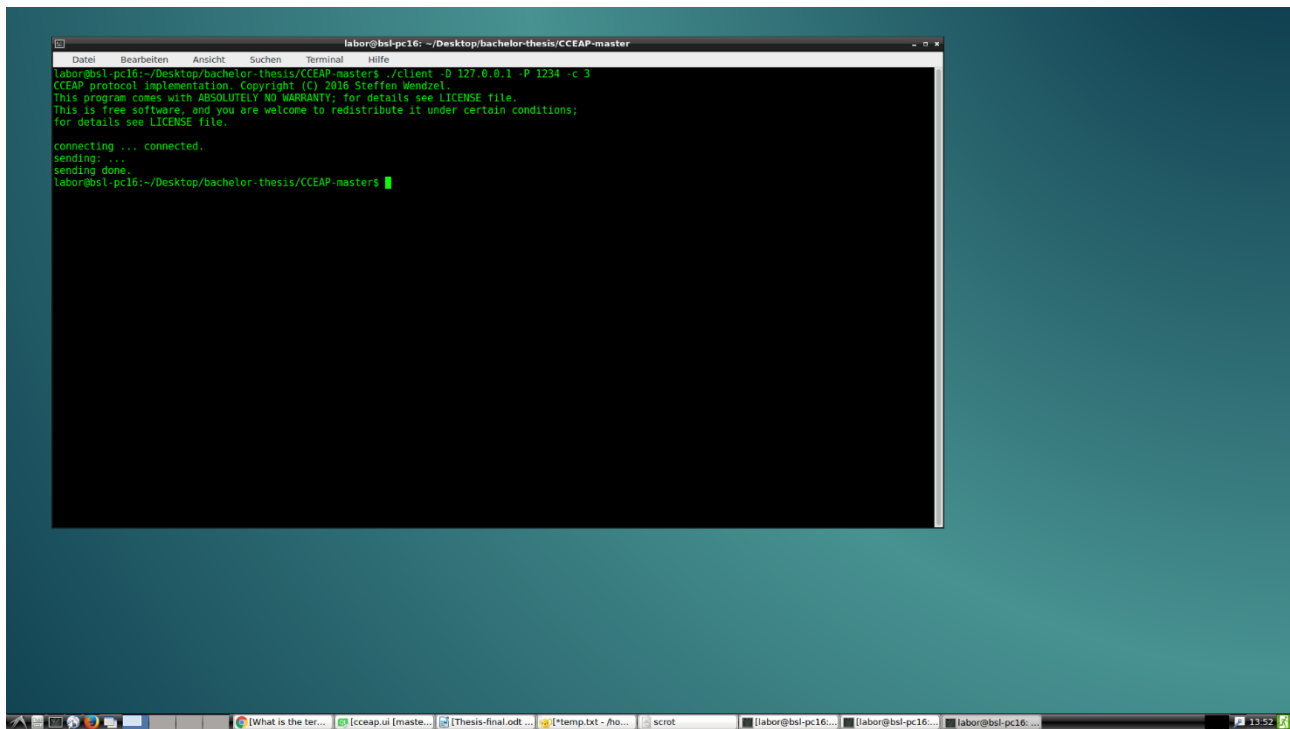
Figure 2: client -h

Figure 3 and figure 4 below illustrates the cceap-client and server started with additional parameters and the respective outputs. In figure 3, the cceap-server is started with a port number, when port number is given on start, the server awaits client's connections, and once an incoming client's connection is received, the server handles the client packets and echoes these packets and their configurations.


```
labor@bsl-pc16:~/Desktop/bachelor-thesis/CCEAP-master$ gnome-terminal
labor@bsl-pc16:~/Desktop/bachelor-thesis/CCEAP-master$ ./server -P 1234
CCEAP protocol implementation. Copyright (C) 2016 Steffen Wendzel.
This program comes with ABSOLUTELY NO WARRANTY; for details see LICENSE file.
This is free software, and you are welcome to redistribute it under certain conditions;
for details see LICENSE file.

CCEAP - Covert Channel Educational Analysis Protocol (Server)
   => version: 0.5.2, written by: Steffen Wendzel, www.wendzel.de
```
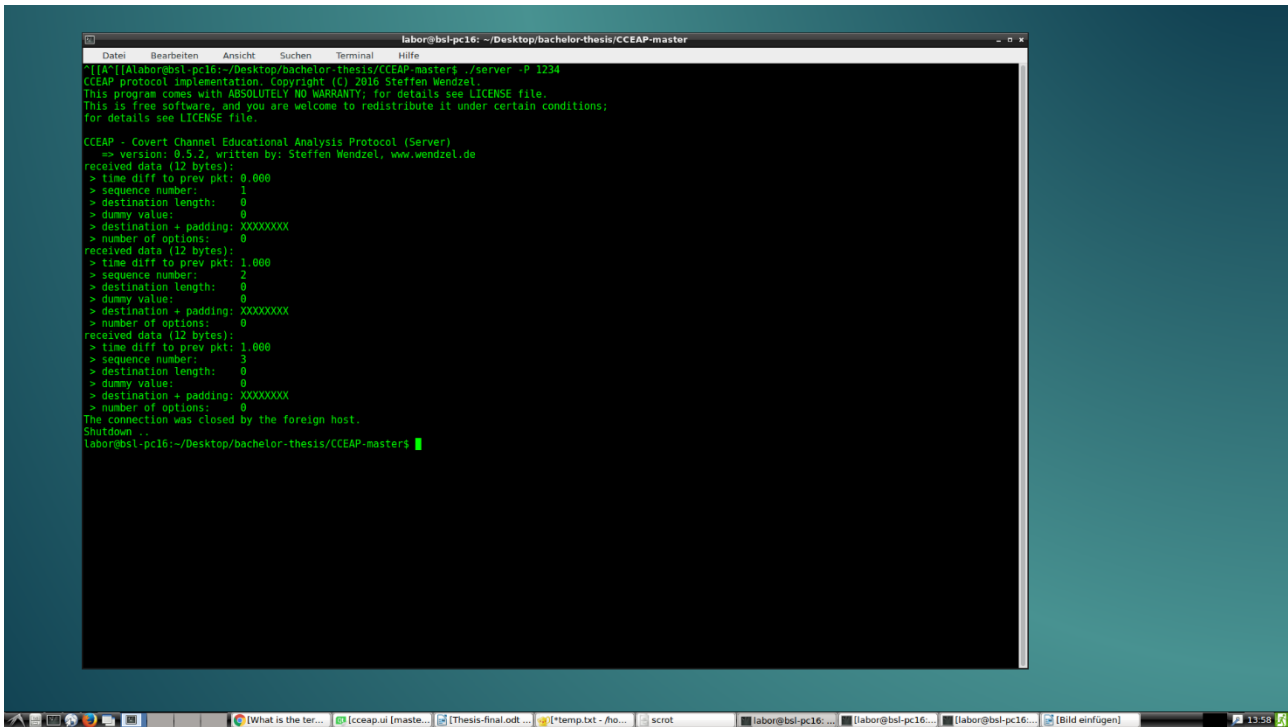
Figure 3: server -P 1234

8

Figure 4: *client -D 127.0.0.1 -P 1234*

Once the server is started on a given port as shown in figure 3 above, it awaits clients from that port. Figure 4 below shows how a client is started with IP-address and port number of the server. The client sends 3 packets to the server, if the option -c 3 is not specified, the client will send the default number of packets 10. The 3 dots after '**sending:**' corresponds to the number of packets been sent. Figure 5 displays the output of the server after receiving the incoming client's packets. In this case the client sent 3 packets only with no additional parameters.

Figure 5: server's output after receiving a client's connection.

1.3 Limitations of the CCEAP Tool

After running the cceap tool, its relevance and support in learning/teaching covert channel is obvious as it provides a practical way to visualize some of the concepts of hiding patterns and covert channels. For the low computer users, or even the novice ones who are not familiar with command lines, it is difficult to use such program efficiently. Below are some aspects of cceap that could reduce the motivation of learners when using cceap to learn hiding patterns and covert channels.
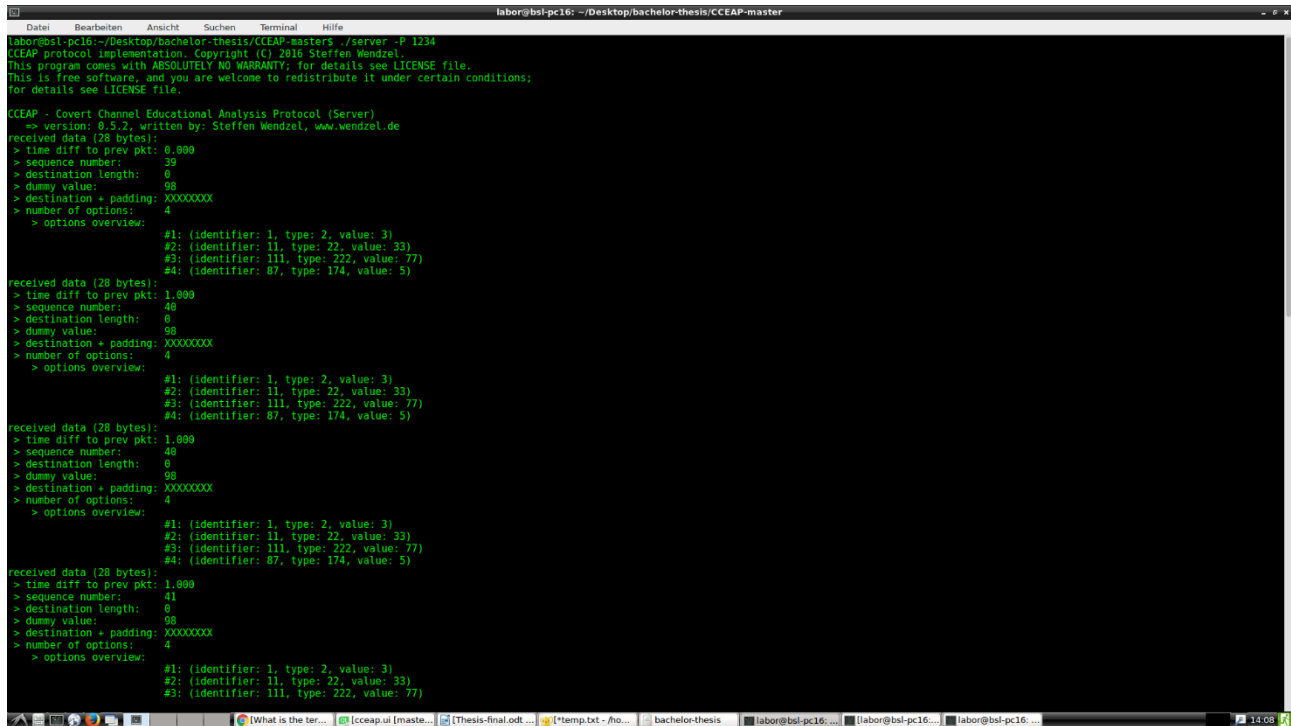
It becomes difficult to follow the flow of the program from begin till end and to interpret the programs output from the command line.

Even advanced users will find it difficult to interpret the output of the cceap server when it is displaying a hundred packets.

The outputs in the server are displayed one line after the other which is simple to understand when only about 10 to 20 packets are being analysed of 20 packets when the screen offers enough room for that. However, when the servers output is a large number of packets and parameters, it becomes strenuous to follow.

Figure 6 below illustrates this problem. It demonstrates the output of the server after server has received an incoming connection. The client sends 10 packets with several parameters, only 4 of the packets can fit on the command line window, scrolling up or down rapidly misleads the user who cannot easily find the link between the packets displayed especially if the user specified the

sequence number for each packet.

Figure 6: server's output to after receiving 10 packets from a client

Another drawback of the command line is the inability to undo and redo when entering parameters. The command line reads parameters lines after lines, so it is difficult to undo/redo un such environment.

The program parameters are not very understandable for novice users even when reading the help section. This tool is easily understandable by experienced users, however casting out the novice who are part of the intended audience.
The command line tool is relatively complex and each system has its own independent command line tool that is used differently from other systems. Thus, having knowledge or being experienced with the windows command line tool does not guarantee the same knowledge could be applied to a Unix command line tool.

Above are only some points that describe the main limitations of the CCEAP. Faced with these, a need to extend features to complete these holes have generated a need for a graphical user interface to interface this command line tool facilitating the use of the cceap.
The section below describes my solution to this, in the form of a graphical user interface for the command line tool.

1.4 My contribution to solve the problem

The above section discussed the little limitations of the CCEAP tool that need to be addressed in some way. Faced with these limitations, I will attempt to provide a durable solution in the form of a **GUI** tool.

This is a GUI (Graphical User Interface) program that serves as interface between the CCEAP command line tool to the user. The user interacts with the GUI which translates the user inputs into CCEAP program parameters, then calls the CCEAP with this parameter and finally displays the output of the CCEAP to the user in a formatted way on the GUI.

Such a tool is quite straight forward in requirements and specifications as it is an interface between users and the program, permitting users to understand the program parameters and facilitating the understanding and interpretation of the program's output.

Below is a list of the specifications that the finished program should fulfil among other specifications.

The program should be highly usable graphical program, that is intuitive to use so as to facilitate users understanding and interaction.

The GUI should also present the cceap's program output in a clear, understandable format to users.

The program should be highly transportable and usable. This means that the program should run on possibly several platforms i.e. Unix, windows MAC OS etc. without requiring any effort. This implies a meticulous choice of programming tools.

The program should be smart and robust in terms of dealing with user's inputs. User's inputs that do not make any sense should not lead to system fail but instead the program should identify the erroneous input and notify the user with the appropriate action to take. In this way helping in the learning process of covert channels and hiding patterns.

The users should be presented the program's output in a format that will be easy to understand and interpret, thereby helping in teaching and increasing the usability of cceap.

2 Design and Implementation

This chapter discusses the implementation phase of the GUI tool. The first section here discusses the program requirements agreed with my professor Steffen Wendzel. The next section presents the proposed program to solve the problem on. Some screenshots of the finished program together with the extent to which they fulfill the stated requirements are discussed together with the explanations of the GUI. The last section is a discussion on my design main decisions regarding why I chose some tools instead of others.

2.1 Program requirements

As agreed with the professor, the GUI tool should fulfill certain requirements which included the following.

- The program should be highly transportable. This means it should run on several environments including windows, mac OS, Ubuntu and others.

- The user interface should be simple and intuitive to use. The users should find it easy to interact with program. The UI controls should provide enough information to the users so that it becomes intuitive what they serve for.

- The code should be of high quality so that it could be published if possible. The program should not crash nor behave unpredictable as a result of user's inputs or program state.

2.2 Screenshots of the GUI

This section presents the program that I created to complement the CCEAP command line tool. This consists of two separate GUI programs that run independently of each other. One is the server GUI that interfaces the CCEAP server and the other is the client GUI.
To each screenshot an explanation is given that first explains the figure, then highlights the ways in which the it fulfills a program requirement stated in section 2.1.
Figure 7 and Figure 8 below demonstrate the outputs of the GUI programs as compared to the CCEAP. Figure 7 shows the output of server GUI on start. When starting, the GUI automatically calls the cceap-server with parameter -h that displays help to the user. Additionally, the GUI displays the IP-address of the computer on which the server is running so that users can easily use this IP-address on the client GUI instead of calling **ipconfig** or **ifconfig** to obtain the same IP-address.
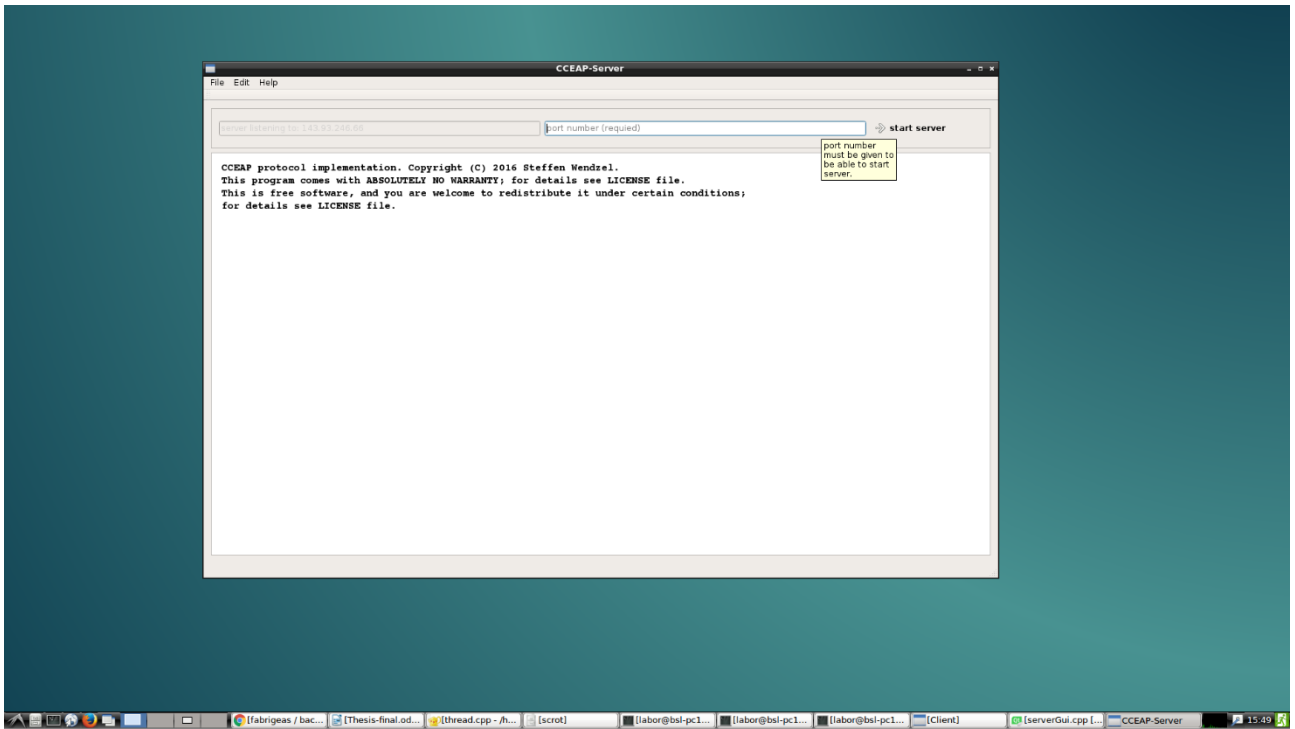
Figure 7: server GUI started

Figure 8 below shows the output of the client's GUI program on start. This also automatically sends parameter -h to the cceap command line program and displays the result to the user.
With these tools, the user doesn't need to type the help parameter to access the help menu. Users can also easily have access to the help menus by clicking on help on the menu bar.
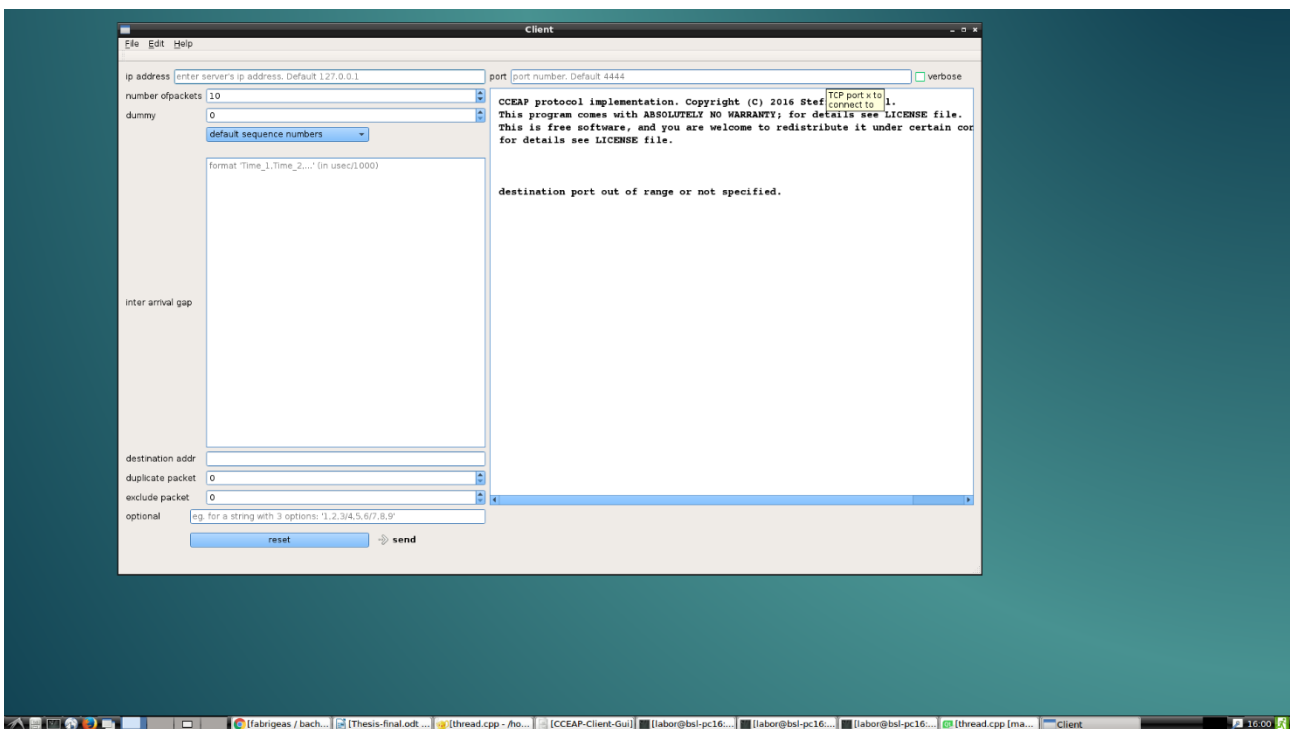
Figure 8: client GUI started

The figures below show the outputs of the GUI programs when users enter the required parameters through the UI controls. Figure 9 shows the server GUI when the user enters a port number and clicks on start server. If the user enters an invalid port number, an error message is displayed providing explanations about the error.
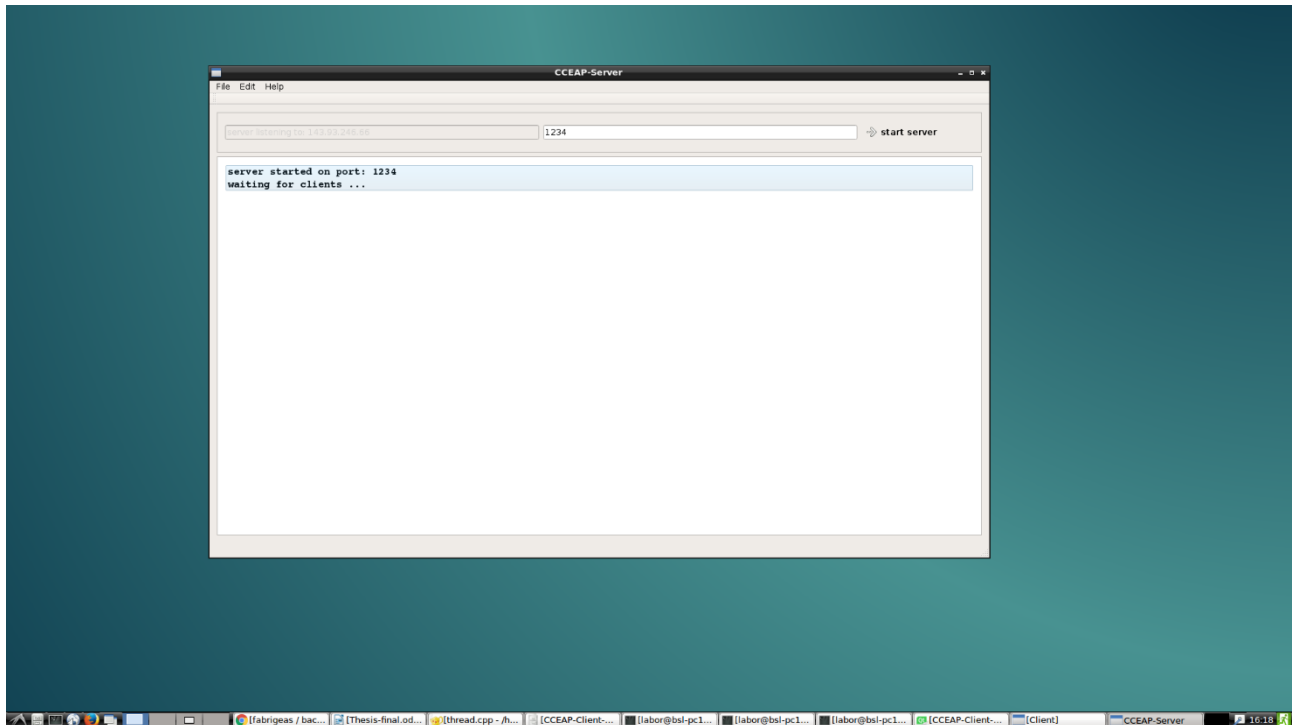
Figure 9: *server GUI - server started with a valid port number*

Figure 10 below shows the output of the client GUI when user enters some parameters. The GUI is very simple and intuitive to use. Every parameter is well labelled and each have a tool-tip that display additional information to the user as to what the UI control stands for (The tool-tip appears when the user places the mouse above the UI control). In figure 10 the client GUI successfully parsed the parameters entered by the client into a valid cceap command line parameter string, and before executing the command, echoes it back to the user so that at any time the user can follow the flow of the program.
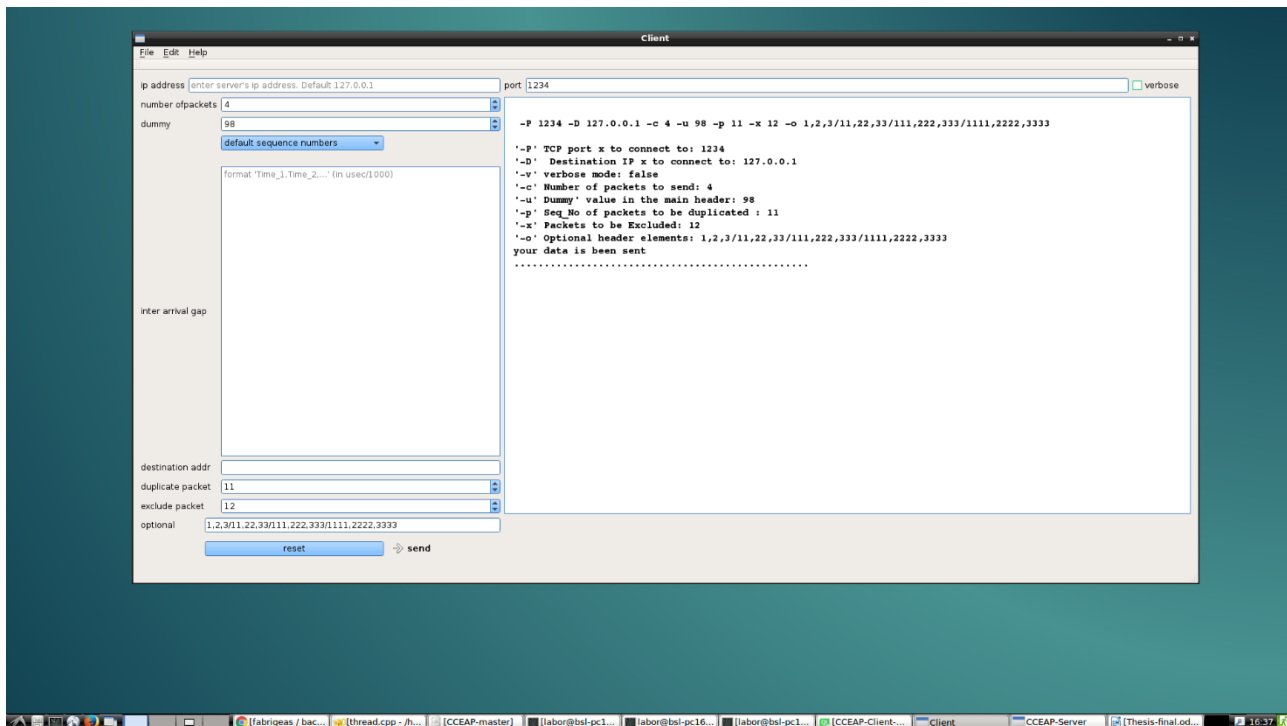
Figure 10*: client GUI-sending packets*

Figure 11 shows the output of the server after receiving an incoming client connection, the output of the cceap-server is first parsed by the server GUI and then displayed to the user, this gives the user the ability to click on each received packets and the packets options. As compared to using the cceap as command line, the GUI delimit each packet such that each is remarkable from the other, this facilitates analysis.
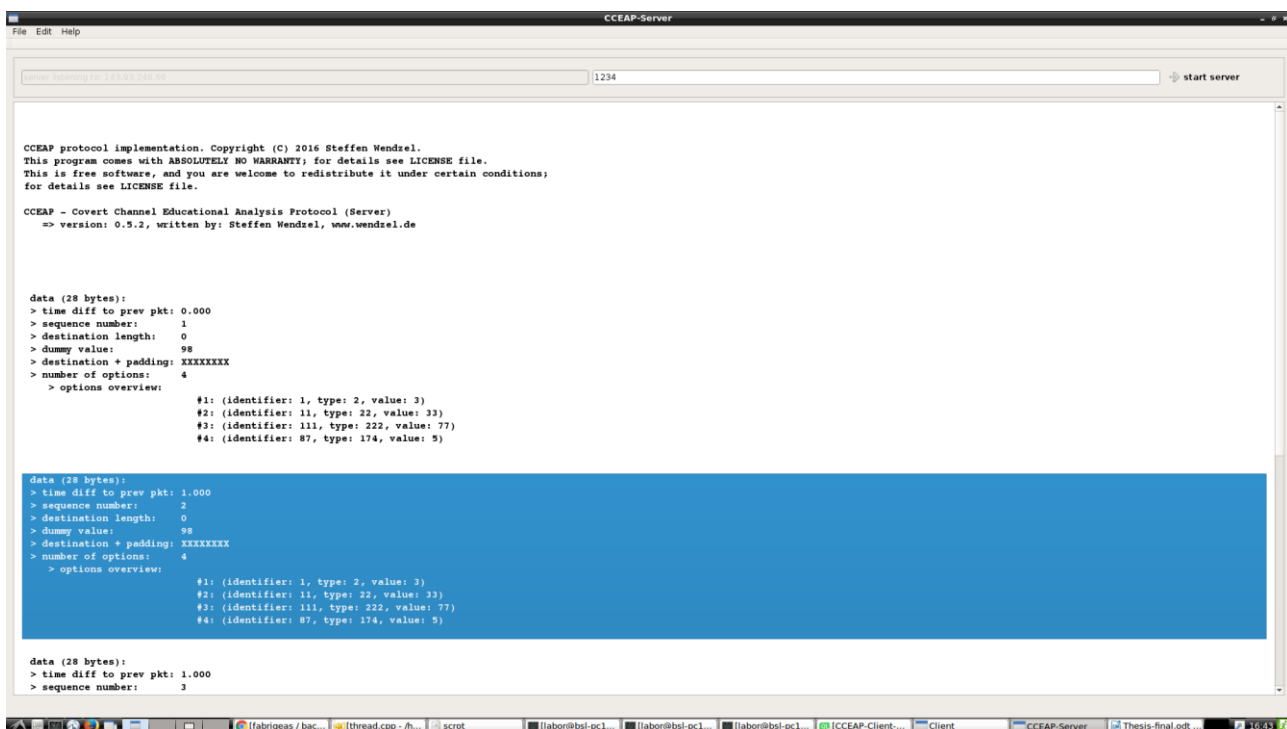
Figure 11*: parsed output of server GU*

16

2.3 Programming Requirements

This section discusses the requirements encountered for the purpose of developing the GUI programs.

To implement the GUI tools, I have made use of a variety of concepts, tools and other programs. Among some of these

QT creator IDE. For designing the layouts and to organize and structure the code I made use of the QT Creator. It is a cross-platform C++, JavaScript and QML integrated development environment. QT creator comes with several useful programming resources including

- a visual debugger

- an integrated GUI layout and forms designer.

- syntax highlighting and autocompletion in editor's features.

- On Windows, it can:

- use MinGW or MSVC

- Microsoft Console Debugger (when compiled from source code).

- Clang.

- C++ compiler from the GNU Compiler Collection on Linux and FreeBSD.

Due to these features, it became intuitive to choose it as the development tool. A more detailed discussion about QT Creator is presented in section 2.5

GitHub

For version control and history management I used GitHub. There are several other tools available such as **sourcetree.** Git is the most widely used version control framework today and is taking over the tasks of versioning, document and clouding programs.

Together with its associated tool GitHub, provides an intuitive approach to even beginners in programming. There are several other frameworks that implement version controls but most of them in one way or another make use of concepts from git. An example of such a framework is source tree [source tree](http://sourcetree.org)

Operating systems

- Ubuntu
- Windows
- Debian
- Mac OS

It was imperative to work with all these operating systems because one of the main

17

requirements of this tool is its portability and transportability.
Transportability in the sense that program, both source code and executable should be easy to export from one platform into another.

As main operating system I used ubuntu 16. and then each time transported the program to my windows and the other platform to generate executables files and test them for distribution.

In the section **deployment** below, I explain how this process of writing the program under platform and the generating a distributed version of the program for another platform.

Deployment is often done in the target platform, i.e. to get the windows executable can only be done on a windows platform and the Unix version on a Unix platform respectively.

2.4 Design decisions

I started writing the program in the Java programming language due to the fact that Java is a highly portable programming language, it is one of the languages that I know the most and I have gathered some experiences through homework and projects in school.

By saying a highly portable language, I refer to the fact that a program written using a Java language on a given system can be exported to other systems without any strenuous operation.

This is because Java programs are not technically compiled to produce a static compiled output, but instead programs written in the Java language are translated from the high-level code of programming language into a java machine language (dynamic code). This java machine is called the Java virtual Machine (JVM).

To be able to transport a program from an architecture (source architecture) to another architecture (target architecture)) the only requirement is the presence of the JVM and JAVAC on the target architecture.

Both the JVM and JAVAC are open-source programs available
https://www.java.com/en/download/

http://www.oracle.com/technetwork/java/javase/downloads/index.html

the system architecture only requires a JVM (Java virtual machine) running in background

- javac JRE

- java JDK

- Eclipse

- Window Builder plug-in for Eclipse

- Windows System

- Ubuntu/any Linux distribution for testing

- virtual box for virtualisation in case only one architecture is available

After more than one week implementing the CCEAP-GUI program in java language, I came to a deadlock and realised that I could not implement such a program using java. The reasons being that the following.

In order to be able to place the UI controls on the screen in the desired positions, the container in which such UI controls would be placed needed an absolute layout such that items(UI-controls) can be placed on the container independent of the positions of the other UI controls already on the container.
However java GUI containers i.e. AWT and swing do not offer absolute layouts but relative layouts, such that an item can only be placed relative to the others, for example in a table or on bordered layout. This was a very serious problem that pulled me back on research.

Another reason for abandoning java in the project was the absence of a designer mode in the most Java programming IDEs. Using the open source eclipse, the only alternative was using plug-ins such as Windows builder, but this plugin was very slow and kept on freezing.

Faced with these problems, I was forced to switch to an alternative language, and after some research I came to choose C with QT creator. This solved my problem of UI designing.

2.5 Qt Creator

Qt Creator is a cross-platform IDE supporting the languages C/C++, JavaScript, QML. QT Creator offers features like a visual debugger, an integrated designer for GUI layout and forms. The editor's features include syntax highlighting and autocompletion.

Qt Creator provides support for building, deploying and running Qt applications for desktop environments including Windows, Linux, FreeBSD and Mac OS.
QT creator also provides programming applications for mobile devices including Android, BlackBerry, embedded Linux devices and several others. Build settings allow to switch between build targets, different Qt versions and build configurations.

CCEAP GUI is developed with QT creator, because as a result of prior research for the ultimate tool to develop the app as it has been described in the problem statement and program specifications, it appears that QT creator offered be the best alternative to develop such a GUI program. As mentioned above I had started implementing the app using java on eclipse and came to a deadlock before coming to QT Creator.
Some of the reasons are explained below.

UI controls are easily placed on the window container as compared to the Java alternative. This is because QT creator allows drag and drop of UI Controls on desired positions of the container, as compared to Java where containers are constraint in Layouts, for example a Border layout only allows a UI control to be placed at the left of the previous controls.

Transporting apps between architectures are relatively simpler on QT as is the case with eclipse. It is to mention here that one of the main advantages of Java over other languages and frameworks is the fact that java code is not compiled for a given output, instead the source code is translated from high to java machine language, and the only requirement to reuse a code on different platforms is the presence of a java machine on the target machine. QT Creator is available under https://www.qt.io/ide/.

3 Deploying the app to windows

This section describes how to create an executable file for the windows platform. In order words, how to run the program on windows in case the application does not provide a windows executable file. The following link offers more detailed information on how to deploy a program developed in QT creator to different target architectures [Deploy an Application on Windows] (http://wiki.qt.io/Deploy_an_Application_on_Windows).

3.1.1 Requirements

- The release version of your app works correctly when you build + run it from Qt Creator

- Qt is installed in C:\Qt\5.2.1\mingw48_32\

- This also applies to

- Qt 5.6 and mingw49_32

3.1.2 Steps (Hard Code)

1. Close Qt Creator.

2. Create a folder to hold the app eg ~\MyApp\

3. cp \QT\Projects\MyApp\MyApp.exe ~\MyApp\

4. cp C:\Qt\5.2.1\mingw48_32\bin*.dll ~\MyApp\

5. cp from C:\Qt\5.2.1\mingw48_32\plugins* ~\MyApp\

6. cp C:\Qt\5.2.1\mingw48_32\qml* ~\MyApp\

7. Test if deployment worked

    1. mv (rename) C:\Qt C:\QtHidden\ (This turns your PC into a clean environment, just like one that doesn't have Qt installed).

    2. ~\MyApp\\MyApp.exe

    3. if it works then perfect

8. Clean-up

    - start ~\MyApp\MyApp.exe
    - while ie running,
    -rm ~\MyApp* and skip erroneous files. (this will delete only unnecessary .dll files)

9. restart ~\MyApp\MyApp.exe

- it works, then the app can be distributed.

4   Problems encountered and Solutions

The program implemented during this project was a simple and quite easy understand. However, the implementation was characterized by lots of problems encountered mostly due to the fact that I was familiar to the programming environment and my lack of experience with such types of programs.

At the end, most of the problems were solved despite taking relatively much time to be dealt with. Among some of the problems that I faced, the following are the most prominent and are those that required much time.

4.1 Layout

I started writing the program in the Java programming language due to the fact that Java is a highly portable programming language, it is one of the languages that I know the most and I have gathered some experiences through homework and projects in school. Although I carried out some research on the portable languages to fit the tasks on hand but at some point, this research was biased by the fact that I already knew java, so I jumped into implementation and spent about 15% of the whole time for this project on a wrong path.
It is extremely difficult to use Java SWING and AWT classes as means of developing complex User Interface programs.

Although the SWING and AWT widgets are designed to create UI programs, they are better appropriate for some tasks than some others, for my task they were not appropriate. These classes are appropriate for UI such as calculator, video player, etc. where a.

For the CCEAP, the available containers apply useless because some items need absolute positioning on well-defined coordinates, whereas AWT and SWING containers only offer Border layout and flow layouts.

4.2 Deadlock

Deadlock is a situation in which it is practically impossible to progress. Executing an application from QT, and making system calls. The second problem and one of the most time consuming problems was making a system call from the GUI program to starting both the CCEAP client and server. I have stayed on deadlock for more than a week trying to run the server and client program from the GUI app using the parameters that the users entered in the latter.
Each of the approaches that I used didn't work and each time I ran the program nothing noticeable happened, at best when I closed the program a debug message said, client program started and is

still running although the GUI app had already exited.

Deadlock with server freezing while waiting for data. After the above problem was solved, I immediately found myself with another deadlock. This happens when the server GUI is started and waiting for clients, after waiting for some time without receiving any client data, the server freezes and only unfreezes once a client has connected and sent some packets. The server GUI then unfreezes and displays the programs output from the CCEAP.

The simple solution has been to create a thread that would execute the desired code section and when is done will signal the GUI thread passing the result to the GUI, then the GUI will display the output of the thread to the user. Below is a code portion that illustrates how I implemented it on QT.

A background threads in the GUIs to be able to run a code portion in background and then update the UI once the background thread has finished running.

This was imperative is solving several problems like the problem of deadlock discussed in section.

```
ServerThread::ServerThread(QString c):command(c) {}

void ServerThread::run()
{

    QStringList list;


    /*QProcess is used to execute an external program.
     * An alternative would be using a system call,
     * but because we still need to process the output of the external programaam
     * we need to start a process instead and
     * waitFirFinish blocks the code to wait for cceapServer to exit and return its
     * output before thisThread can proceed.
     * This is required because we need to process the output of the cceapSer-
ver.
     */

    QProcess cceapServer;
    cceapServer.start(command);

    //forces this.Thead to wait for output of cceapServer
    cceapServer.waitForFinished(-1);

    /*read all cceapServer's outputs
     * both readAllStandardOutput and readAllStandardOutput
     * are returned in 'readAllStandardOutput'
     * so no need to read both of them
     */
    QString stdout = cceapServer.readAllStandardOutput();
    //QString stderror = cceapServer.readAllStandardOutput();


    /*The output of cceapServer is a string, howver the GUI needs alist of pa-
ckets
     * of list of error to display to users, so the stdout string must be split
into a StringList
     * before been returned.
     * In the cceapServer output, each packet begins with received Data ...
```

24

```cpp
     * so we use such pattern as delimiter to split the String into an String
list
     *
     */
    QRegExp rx("received");
    list = stdout.split(rx);



    //return the processed data(StringList) to the GUI then exit
    emit signal(list);



}



/** starts a background
 * the thread will run the command and return a QStringlist
 * this stringlist will then be displayed by the caller
*/
void ServerGui::execute(QString command){

    stringList.clear();
    ServerThread *serverThread = new ServerThread(command);
    QObject::connect(serverThread,SIGNAL(signal(QStringList)),this, SLOT(dataRe-
ceivedFromServer(QStringList)));
    serverThread->start();
}
```

4.3 Live communication between cceap GUI

The last but not the least problem was that of live communication between the CCEAP GUI and the CCEAP app. This is not considered as a problem because it was not included in the programs backlog. It was just intended to be a feature of the initial GUI app but instead it is an extension that I have left behind for the future students that will continue the development of this application.

The feature consists in that; the server GUI should display live output from the server program as it is bean outputted. Currently, the GUI displays the programs output once the cceap server has exited. It would be a nice feature to have the cceap server's output lively displayed on the GUI. I am not sure that it is possible to have such output but this research was out of the scope of my thesis, so I left this task for the future developers of the CCEAP.

Nonetheless, I have spent considerable time researching on how to implement this task.

Again, the solution to this was the use of a thread that ran in background and continuously emitted signal to the GUI thread with resulting data, the GUI thread then received the signal and updated the data on the display. This solution sounds very simple however it was not that obvious during the implementation as it did not work on several tries.

4.4 My remark

Despite all the planning that I went through, I have come to learn that, practice is different from theory and contingency is always the best plan when experience is unavailable. The program is relatively simple, but this appears so only after solving all the problems. Arriving at such a simple program layout required a lot of trials and errors.

During and after the implementation, I have come to identify some features that could be added to this GUI tool as well as some of the requirements that I could not meet. Requirements that could not be met are partially due to the fact that a modification to CCEAP needed to be done in order for the requirement to be feasible. These features and unmet requirements are discussed in chapter 5 (Extensions).

5 Extensions to the application

The problem statement of this thesis requires the development of a GUI tool that provides interface to interact with the CCEAP command line program.
The basic requirements are the simplicity and portability of the program. Portability in terms of the possibility and ease with which the program can be ran on different operating systems and architectures.
The requirements of the program are quite clear and the required features have totally fulfilled.

However, during and after the implementation, I have come to identify some features that could be added to this GUI tool. These features I have called extensions as it is often done in computing to mean features that could be added to an existing program to extend the programs functionalities. Below I have listed some of these functionalities, and these are intended to be developed in the future by students who will continue the development of CCEAP.

5.1 Export session

The first feature will be an Export command. This could be extended to the client's GUI so that once a session with all input parameters for a purpose will be used, these settings could be exported to an external file in JSON, XML, CSV, oData or any other format, so that next time the user do not need to re-enter such parameters but can open the file in which these settings were saved and directly copy from there into the GUI.
This will increase the overall usability of the program.

5.2 Import session

As a further extension to the above extension, could be the introduction of an import feature, this permits users to directly import settings from a file into the GUI program. This is very practical when for example a client starts an analysis session with 100 packets giving each packet a sequence number. Even though these 100 sequence numbers might have been saved into a file, typing them into the program is quite strenuous and time consuming. Export and Import features have been separated because each export might not absolutely require and import. The user might just need to save settings into a file without forcedly needing to import them.

5.3 Live communication

Live communication between UI tools and programs: This feature should permit data to be displayed on the GUI while the cceap program is performing actions. Currently, the GUI starts the CCEAP program with the parameters entered by the user, then only display the returned output of the cceap after the latter has exited.

The idea here is to display the cceap's output while the latter is running, not simply displaying the latter's output after it exits.

I have not researched on this much because it was out of the scope of my thesis, so I left it behind for future developers.

5.4 Load Test

A further feature but this time the starting point will be testing the GUI and CCEAP on load tests. The tests that were carried during and after implementation were functionality tests to ensure that the programs behave as expected independent of user's inputs and systems architectures.

All tests (both systems and functional) returned positive as the program recognises user's input errors and handles them as expected. After all tests, no know bugs have been identified.

The task here will be to test both cceap and the corresponding GUI tools on their behaviours when faced with overloads for example sending a thousand packets.

These have not been tested because thousand packets as parameters go out of the scope of CCEAP as students might not need to analyse a thousand packets in a session while learning covert channels. The necessity to test appear now because as CCEAP becomes widely used, several users might have to connect to a single server, the behaviour of the server in response should be tested.

6 Conclusion

The aim of my thesis has been to implement a GUI (Graphical User Interface) program as a tool to complement an existing program (CCEAP tool).
In the introduction, I have explained all useful terms and concepts related to CCEAP and the reasons and purpose of the CCEAP.

The importance of this work lies not only on the program, but also on the application of the concepts that I have acquired during my undergraduate studies. I would like to emphasize on the fact that the application developed in this project is only for an academic purpose, a lightweight GUI-tool with the sole purpose of providing a Graphical User Interface to the already existing command line program. It is not intended for commercial purpose but instead to serve for academic purposes specially to contribute in learning/teaching network steganographic hiding patterns.

The finished program is relatively simple in terms of layout and even the code structure looks simple and easy, however this is just the finished program. This appears so because it is the finished program, arriving at such simplicity required a lot of research and programming. It is only after several tries, debugging and code refactoring that I have been able to produce such simplicity. My lack of experience and my unfamiliarity with QT creator only served as counter effort, but finally I came out with a program that partially if not totally fulfils its requirements and in the processes, I acquired much knowledge and experience with QT creator.

7 Abbreviations

- CCEAP: Covert Channel Educational Analysis Protocol

- CMD: Command Line tool

- CSV: Comma Separated Values

- Enet: Ethernet

- GUI: Graphical User Interface

- HTTP:

- IDE: Integrated Development Environment

- IP: Internet Protocol

- Javac: java compiler

- JDK: Java-Development Kit

- JRE: Java-Runtime Environment

- JSON: JavaScript Object Notation

- JVM: Java Virtual Machine

- oData: Open Data source

- OS: Operating System

- OS: Operating System

- QML: Qt Meta Language/Qt Modelling Language

- SDK: Software Development Kit

- TCP/IP: TCP+ Internet Protocol

- TCP: Transmission Control Protocol

- TCP: Transmission Control Protocol

- UI: User Interface

- UDP: User Datagram Protocol

- XML: Extensible Markup Language

8 Bibliography

[Hidden and Uncontrolled - On the Emergence of Network Steganographic Threats]
(https://www.researchgate.net/publication/263773592_Hidden_and_Uncontrolled__On_the_Emerg
ence_of_Network_Steganographic_Threats).

[C++GUI Programmingwith Qt 4Jasmin BlanchetteMark Summerfield ISBN 0-13-187249-4Text
printed in the United States on recycled paper at Courier in Stoughton, Massachusetts.First printing,
June 2006](http://www-cs.ccny.cuny.edu/~wolberg/cs221/qt/books/C++-GUI-Programming-with-
Qt-4-1st-ed.pdf)

[Network Information Hiding Patterns](http://ih-patterns.blogspot.de/p/introduction.html)

[Information Hiding in Communication Networks: Fundamentals, Mechanisms, Applications, and
Countermeasures, Wojciech Mazurczyk, Steffen Wendzel, Sebastian Zander, Amir Houmansadr,
Krzysztof Szczypiorski ISBN: 978-1-118-86169-1 296 pages April 2016, Wiley-IEEE Press]
(http://eu.wiley.com/WileyCDA/WileyTitle/productCd-1118861698.html)

[An Educational Network Protocol for Covert Channel Analysis Using Patterns
(http://dl.acm.org/citation.cfm?id=2989037)

[Pattern-Based Survey and Categorization of Network Covert Channel Techniques]
(https://www.researchgate.net/profile/Steffen_Wendzel/publication/263048788_Pattern-
Based_Survey_and_Categorization_of_Network_Covert_Channel_Techniques/links/54d271830cf2
b0c614697b52/Pattern-Based-Survey-and-Categorization-of-Network-Covert-Channel-
Techniques.pdf)

[User CCEAP command Line tool] (https://github.com/cdpxe/CCEAP)

[Download Javac] http://www.oracle.com/technetwork/java/javase/downloads/index.html

[Download JVM] https://www.java.com/en/download

[QT Creator] https://www.qt.io/ide/

[Source tree] (http://sourcetree.org)

9 List of figures