Coursework Declaration form

I declare that the work contained in this assignment is my own, unless otherwise acknowledged.

No substantial part of the work submitted here has also been submitted by me in other assessments for any other assessed course, and I acknowledge that if this has been done an appropriate reduction in the mark I might otherwise have received will be made.

The submission will be passed through the Turnitin system for plagiarism testing, so please ensure your work is thoroughly cited/referenced and does not contain 'copy and paste text'. All cases of suspected plagiarism are treated seriously and could affect the outcome of your final mark.

I have used a proofreader, paid or unpaid, to support the submission of this assignment:

[✓ YES]
☐ NO

The University expects all proofreaders to comply with its policy in this area. By ticking 'yes', you confirm that the proofreader was made aware of and has complied with the University's proofreading policy, found here https://www2.warwick.ac.uk/services/aro/dar/quality/categories/examinations/policies/v_proofreading/

Signed:

1925723 1914526 1935098 1911439

MODULE TITLE:	Software Development and Security
MODULE CODE:	WM145
WORD COUNT:	5072
UNIVERSITY No:	1925723 1914526 1935098 1911439

WMG
THE UNIVERSITY OF WARWICK

Software Development and Security

Frankenstein’s Toolkit Report

By
Harry Heather
Henry Williams
Jack Hainsworth
Murat Saglam

1.0 Introduction

1.1 Purpose

The purpose of this specification is to provide a guideline of how this program was created, intended for those wanting to recreate it, or any of the individual components. It also details the prerequisites needed to operate the program.

1.2 Scope

This specification is for the program “Frankenstein’s Toolkit”. The program was designed for educational and testing purposes, with the end goal of learning more about network standards and security, as well as basic cybersecurity processes.

The toolkit contains four tools: a Denial-of-Service (DoS) tool, a Packet Analyser, a PCAP Fabricator and a Steganography tool. While these tools have the potential to be used for malicious purposes, care has been taken to assure that the tools will not be used in any immoral way.

The program is intended to be deployed over the internet, accessible via a web front end which will provide the user with a menu containing the tools. User accounts are used to secure the toolkit from unauthorised access.

1.3 Overview

The following report is broken down into the following sections:

- 2.0 Overall description - a more detailed explanation of what the program does
- 3.0 Specific requirements - information a user needs to operate the toolkit
- 4.0 Project Plan - how the toolkit was created
- 5.0 Conclusion - final thoughts on the task and improvements that could have been made

2.0 Overall Description

2.1 Product Functions

The functionality of the toolkit is separated into four tools:

The DoS tool sends a stream of data packets from a spoofed IP address, towards a target IP address. On setting up a network listener, a user can see the packets leaving their network using the header parameters specified in the DoS tool.

The PCAP analyser deconstructs a packet capture file and presents information from the captured packets in an easy-to-read format. For each packet in the PCAP file, it displays the source and destination IP address, source and destination port, the protocol used to transmit the data and other relevant information.

The PCAP fabricator is able to create a single packet capture file with a single data packet. The data packet will have the attributes of the information that the user has entered. This packet then can be analysed using a text editor in order to understand the structure of a network packet.

The steganography tool enables a user to upload / input their own text message into a carrier image file, which is then encoded into the LSB of every pixel. This means that a user can upload an encoded message and the steganography tool is able to analyse the image and decode the message.

2.2 Constraints

The primary constraint on designing the toolkit with the potential for malicious activity. The DoS tool, for example, could accidentally flood a public network with traffic if the wrong destination IP address is entered. To prevent any wrongdoing, the tools were constructed in isolated VMs with no internet access.

The toolkit was designed to have as few factors limiting its design as possible. Part of the reason the toolkit has a web front end is because it offers considerable freedom as an interface for the tools (e.g. supporting multiple programming languages).

2.3 Assumptions and Dependencies

DoSer Dependencies

Library requirements:

- arpa/inet.h
- errno.h
- netinet/tcp.h
- netinet/ip.h
- stdio.h
- stdlib.h
- string.h
- sys/socket.h

PCAP Analyser Dependencies

- Libcap.c

PCAP Creator Dependencies

Library requirements:

- sqlite3
- Sys
- Binascii
- socket

Front-End Dependencies

Library requirements:

- Python3
- flask
- flask_login
- subprocess
- os
- flask_bootstrap
- flask_sqlalchemy
- flask_migrate
- flask_marshmallow
- flask_wtf
- Wtforms

Front-End Assumptions

- First of all in the directory that contains toolkit.py perform the command:
  - export=FLASK_APP=Toolkit.py
- You are then able to use the following command to run the front-end:
  - flask run
- Due to the nature of how the DDOS.C program has been created, flask run must be run as a root user on your system.
- Due to the front-end not being deployed publicly due to security concerns raised by Jack and personal choice. The directory based items have been configured for my personal VM therefore in order to run the front-end fully the following items will need to be configured for your personal machine:
  - The following variables within config.py
    - UPLOAD_IMAGES_ENCODE_STEGNO
    - UPLOAD_MESSAGES_ENCODE_STEGNO
    - UPLOAD_IMAGES_DECODE_STEGNO
    - PCAP_UPLOAD_DEST
    - PCAP_DOWNLOAD_DEST
    - ENCODED_DOWNLOAD_DEST
    - DECODED_DOWNLOAD_DEST
    - PCAP_OUTPUT_DEST
  - The following variables within ImageStegno.py:
    - TextFile (Line 16)
    - self.Size (Line 20)
    - self.image (Line 28)
    - NewImage.save (Line 80)
    - NewImage.save (Line 110)
    - NewImage.save (Line 135)
    - EncryptedImage (Line 140)
    - DecodedFile (Line 219)
  - The following variables within PCAPCreator.py
    - path (Line 61)
    - conn (Line 110)
  - The following variables within PCAPRead.c
    - Char path[150] (Line 62)

Steganography Dependencies

- Pillow (PIL)
- binascii
- os
- flask

3.0 Specific Requirements

The main goal for the cyber tool kit is to provide portable essential cybersecurity tools, that is why one of our aims with this program is to have minimal requirements for start-up and throughout. However in order to be able to access the cyber tool kit, you will need a device which has access to the internet, however it's not currently being hosted. The website is designed to be accessed on desktops/laptops however other devices still have the functionality to access the site, but their user experience will not be as good as using a desktops/laptops.

3.1 External Interface Requirements

All programs are required to have an input output process, this can be as simple as pushing a button for the input process, and as complicated as booting up an operating system for a machine as the output. In this section of the report we will be analysing in-depth the inputs and outputs of the program, and in some cases the required preconditions in order to make sure the application works correctly, sometimes referred to as input sanitation.

3.1.1 Hardware Interfaces

As mentioned in 3.0 there are minimal hardware requirements in order to have ease of accessibility to a wide range of audiences. The hardware requirements is as follows:

- An electronic device that has a correctly configured NIC (Network Interface Card)
- A hardware device that provides connectivity to the internet, this can include:
    - A dongle
    - A router
    - A device that has access to the internet with hotspot capabilities
    - An ethernet cable (if the device has the appropriate port to connect to)
- A device that provides a display e.g monitor
- A device that provides character input e.g keyboard

There are no specific requirements for any of the tools within the tool kit, therefore these hardware requirements are the same for all of the tools.

3.1.2 Software Interfaces

Our application is divided into four separate tools for the tool kit. They will be abbreviated as such:

- X.x.x.1 for the DoS tool
- X.x.x.2 for the Pcap fabricator tool
- X.x.x.3 for the Packet analyser tool
- X.x.x.4 for the Steganography tool

Each tool has a large range of inputs and outputs, therefore they're divided as such.

All important input sanitisation is done by the front end, refer to the forms.py file for the specific sanitisation.

3.1.2.1

Input table

Variable name	**Input**	**Sanitisation**
Source IP Address	Spoofed address that the packet is sent from	Must be a valid IP address - four octets of dotted-decimal, each no greater than 255
Destination IP Address	Target to which the packets are sent	Same as above

Output
The tool outputs a success message once a packet is sent.

3.1.2.2

Input table

Variable name	**Input**	**Sanitisation.**
Name of the pcap file (taken as an argument from the command line)	To save the pcap file to the users preference	No sanitisation
message	User's message for the pcap file	Not greater than 31 bytes of information
port	The port the packet is going to be sent to	Has to be an integer
src_ip_address	Ip address of the source of the pcap file	Valid IP address under the requirements of the socket module
dst_ip_address	Ip address of the destination of the pcap file	Valid IP address under the requirements of the socket module
src_mac_address	mac address of the source of the pcap	Valid MAC address

	file	
dst_mac_address	mac address of the destination of the pcap file	Valid MAC address

Output
The output is a single pcap file which contains a single packet with the properties of the aforementioned variables.

3.1.2.3

Input table

Variable name	Input	Sanitisation.
file	Name of the pcap file	Not empty, is a .pcap file.

Output
The output is a .txt file containing packet information and general statistics.

3.1.2.4

Input table

Variable Name	Input	Sanitisation
TextFile (Line 209)	The sanitised filename for the text file the enters on the web form on /ImageEncode passed via routes.py to Encode function	FIle name sanitised via the secure_filename function to avoid possible modification of server side files by a front-end user
ImageFile (Line 209,216)	The sanitised filename for the image file entered on the web form on /ImageEncode and /ImageDecode passed via routes.py to their corresponding functions	File name sanitised via the secure_filename function to avoid possible modification of server side files by a front-end users
TextFile (Line 16. This should not be confused with the filename this is the stream caused by using open on the uploaded text file)	This is the actual text file data uploaded by the user that is read into the variable self.RawData and turned into binary via the function GetBinary() in the text object.	First of all the front-end will sanitise against any non .txt file inputs made into the front-end. And also forces users to enter a txt file in order to run the stegno tool. While the contents are not

		checked its size is compared to the size of the image uploaded to check whether the message can be fit in the image (If encoding)
Self.image (Line 28)	This is the image file object of the users uploaded image file that is then manipulated.	The front-end will sanitise any files that are not of the format png,bmp, jpeg this is to ensure that no file types uploaded use lossy compression (corrupting the message). ImageStegno.py itself will first of all check that the image is large enough to fit the message (If encoding) and it is of a valid image type to be encoded / decoded.

Output
Depending on the function used the encode function will save a new image file called Encoded(users image file name) that will contain the user's message encoded within it. The decode function will produce a .txt file named Decoded(users image file name) that will contain the decoded message within it. Else if it errors it will produce a corresponding flash message explaining the error

3.2 Software Product Features

3.2.1 Doser (Harry)

Specific requirements for the function

To fulfil the requirement, the DoS tool must:

- Take a source and destination IP address
- Sanitise the inputs
- Fill the TCP and IP headers with relevant information
- Create a network socket
- Send the dummy packet in an infinite loop

Use Cases of the app

The DoS tool was designed for education and network testing. A user learning how to prevent a DoS attack could utilise the tool to test their network's security and performance, in a controlled environment.

Intended developments for future iterations

Potential upgrades include:
- Making it a "distributed" denial-of-service tool
- Allowing all header parameters to be taken from the user

3.2.2 PCAP Fabricator (Murat)

Specific requirements for the function

In order for the application to be successful it will need to be able to:
- Get the users inputs
- Sanitise the inputs so they are suitable to a pcap file
- Append the inputs correctly to a text file
- Save the text file as a pcap file
- Return the pcap file to the user with the appropriate changes.

Use Cases of the app

The pcap file can be opened using a text editor, this can provide educational features since a user can access a low-level interpretation of a data packet. This can help solve problems that will be rather strenuous to solve such as "Can I insert mac addresses to a UDP data packet?".

Intended developments for future iterations

Due to time constraints and limited knowledge of pcap files, there was only a limited amount of customizability that I was able to add. Therefore for future iterations I will like to add different:
- File formats
- Unique TTL (time to live)
- The source port
- Different type of IP addresses
- Time manipulation for the packets

These additional features can provide functionality that will be more useful to a larger audience since the features open avenues to experiment with packet spoofing, which appeals to a larger audience.

3.2.3 Packer Analyser (Henry)

Specific requirements for the function

1. Ability to take in a .pcap file.
    a. Ability to check that the file is not empty.
    b. Ability to check that a .pcap has been specified in the command line.
    c. Ability to check that the file contains packets.
2. Ability to process each packet.
    a. Ability to extract wanted information from each packet, including:
        i. Ip source and destination.
        ii. Port source and destination.
        iii. protocol(s).
    b. Ability to convert common ether types from numbers to names.
3. Ability to process the information extracted from the packets.
    a. Ability to know if an IP address has sent/received a packet before and if so know how many packets that IP has sent.
    b. Ability to know if a port address has sent/received a packet before and if so know how many packets that IP has sent.
    c. Ability to order Ip addresses by packets sent.
    d. Ability to order Ip addresses by packets received.
    e. Ability to order Port numbers by packets sent.
    f. Ability to order Port numbers by packets received.
4. Ability to output information in a transportable method.

Use Cases of the app

This app/toolkit would be used if a user had a .pcap file but no other program to process it, an online program allows users who quickly need a pcap processed, but don't look at pcaps enough to want to install a program for it. This program will also give a very fast overview of a file, which is helpful for when specifics are not needed.

Intended developments for future iterations

To improve the program further, more information could be extracted from each packet, such as packet length, frame number and packet info. Along with this more ether type translation could be added for those more uncommon packet types.
Another beneficial feature would be to improve the connection between the program and the frontend. Giving the user more options to take in the output, such as graphing the results would improve the readability of the output.

3.2.4 Steganography (Jack)

Specific requirements for the function

1. Ability to take and handle uploaded image files
    a. Ability to Check that an image uploaded will be able to fit the message
    b. Ability to Check that an image is of correct format and type i.e (Not the type of 1 or an unknown type)
    c. Ability to store information about the opened image such as its size and format
2. Ability to take and handle uploaded text files
    a. Ability to check that the message will fit the image (i.e it's not too big)
    b. Ability to check that the message file is a .txt file
    c. Ability to extract the size of the file as well as the message and the binary form of that message
3. Ability to encode images and messages
    a. Ability to create a new image in order to store the message in a near identical copy of the original image
    b. Ability to get data from each pixel from the image uploaded by the user
        i. Ability to get the RGB values from each pixel
    c. Create a bitstream from the text file message as well as including a delimiter to show the end of a message
    d. Ability to replace the LSB value of each colour bit
    e. Ability to place the pixel within a new image and then save the new image
4. Ability to decode images
    a. Open the Image and view the LSB of each color value and record as the bitstream
    b. After every LSB see if you have reached the message delimiter after converting the binary to string

Use Cases of the app

The cases where this app/toolkit would be used would be if a user wanted to send a private message but didn't want to use traditional means such as text or message applications a user could send an image to whoever he wants with the message encoded in the image and if that message was somehow intercepted no one would be able to read the contents due to them being encoded within the image. This then would allow the receiver to use the front-end to upload the encoded image and retrieve the message.

Intended developments for future iterations

A feature that was looked at but not added was the ability to encode and decode sound therefore a future development opportunity would be the ability for users to encode and decode messages using .wav files.

Another feature I would develop/expand upon is the performance of the tool. I have noticed that with large images due to the method of creating a new image and copying the pixels from the original image I noticed a long run time. I would likely in the future instead of creating a new file look to simply replace the bits of the existing file cutting down on the performance cost needed.

Finally, I would like to add support for more image modes in order to expand the availability and reliability of the tool.

3.2.5 Frontend Interface (Jack)

Purpose

The main purpose of the front-end was to provide a friendly and interactive interface for the user to use the tools. I aimed to avoid a user having to use the terminal to run the tools making it more interactive. It also acts as the glue to link all the tools together so it's not essentially a batch of programs. I also did this so the user would not have to worry about installing certain libraries etc to run the programs and also make it more accessible to those with poor machines as some of the programs especially the steganography can put a large strain on performance.

Specific requirements for the function

1. Allow users to register to the front-end
    a. Ability to create a route to a register page
    b. Ability to get the registration form from the forms.py file
    c. Ability to display the form via a html template from templates folder
    d. Ability to store the users information within the sqlite database
2. Allow users to login to the website
    a. Ability to get login form from the forms.py file
    b. Ability to get Display template from templates
    c. Ability to Validate whether the user exists in the SQLite db
    d. Ability to actually login the user and remove the anonymous status on the user
3. Prohibit users to access certain pages such as tools if they are not logged in
4. Create a navigation bar to allow for easy navigation of the website instead of having a user have to type /login etc.
5. Templates folder to store HTML templates
6. Forms program to handle all form information for the front-end
7. Creation of SQLite database in order to store necessary information
    a. Models program to hold tables for the database file
8. Ability to logout users
9. Implementation of the PCAPAnalyser program to the front-end
    a. Make it so you have to login to view the page
    b. Ability to input a pcap file via a webform
    c. Ability to run the C program from the flask front-end

    d. Ability to save the file on the host machine
        i. Ability to validate filename for security
    e. Ability for the user to download a file showing the output
10. Implementation of the PCAPFabricator program to the front-end
    a. Make it so you have to login to view the page
    b. Ability to input necessary information via a webform
    c. Save the information to the SQLite database
    d. Ability to Download the PCAP file being created
11. Implementation of the Steganography tool to the front-end
    a. Ability for the user to choose whether they want to encode or decode an image
    b. Save a .txt and or image file to the system
    c. Run either the encode or decode function of the tool depending on the user
    d. Ability to download the encoded or decoded message created
12. Implementation of the DDOS tool to the front-end
    a. Ability for the user to input the source and destination IPs
    b. Ability to run the DDOS command from the front-end
    c. Ability to output the result of the DDOS tool to the front-end
13. Ability to implement bootstrap form templates as well as navigation bar etc.

Use Cases of the app

The cases where the front-end would be used is if a user wants an easy to access and useful toolkit in order to perform various cyber related actions. It would also be used mainly by people who don't want to install an individual tool on their system or lack the required libraries to use certain tools / necessary processing.

Intended developments for future iterations

Certainly one area I would develop further is general aesthetics of the website and the web design as while it functions it does not look great and leaves much to be improved. This could be improved via static CSS files and javascript showing elements such as file upload progress.

Another area I would certainly like to develop further is the profile currently it is very bare showing only the users name. I would like to expand this with a profile picture and other information that the user could enter. And the ability to view other users profiles and possibly implement a message/social system to the website that allows for users to communicate amongst each other. This would likely be done via the implementation of sockets however further research is required

3.3 Software System Attributes

3.3.1 Reliability

While a program can never be called “bug-free”, rigorous testing of the toolkit has established that none of the tools will consistently fail under normal running conditions. In a real-world scenario, evidence of reliability would be further compounded by activity logs.

3.3.2 Availability

The toolkit excels in the context of availability. Being accessible via the internet, the toolkit is available for use on any device with a web browser, for as long as the toolkit is hosted online.

Cloud hosting providers offer a certain guarantee of the mean-time between failure of their hardware. Depending on the provider, the services they host are guaranteed to be available for around 99% of a year; a far greater estimate than if the service were hosted locally.

3.3.3 Security

User accounts are implemented in the front end to gate-keep the toolkit from unauthorised access. The addition of activity logs would also offer a method of tracking which user performed which task. Cloud-hosting of the toolkit means that the hosting servers are protected by a robust third-party firewall.

3.3.4 Maintainability

Each program made for the toolkit is separated in its own file. This along with good commenting throughout the code allows for easy review and maintenance.

3.3.5 Portability

Porting the webpage and backend from host to host shouldn’t be too large an issue, with dependencies and set up instructions listed in 2.3, recreating the current setup should not be too difficult, however with DOS.c requiring root privileges to function, some official hosting companies may be reluctant to host, to fix this issue changes would need to be made to DOS.c to not require root privileges to run.

4.0 Project Plans

4.1 Project Requirements

4.1.1 Task List - General

1. Work on Individual programs, ref 4.1.2, 4.1.3, 4.1.4, 4.1.5.
2. Complete frontend of the webpage (Jack).
    a. On completion of individual programs, implement into the webpage (Jack).
3. Ensure programs run smoothly when used from the webpage.
4. Complete documentation.

4.1.2 Task List - DoSer (Harry)

1. Research methods of denying a service and how can they be realistically created
2. Confer with the team to make sure the tool can be used via the front end (i.e. can use both C and Python).
3. Reinforce learning with details such as:
    a. How sockets are used to send data
    b. How IP and TCP header parameters are set in the netinet libraries
4. Assure proper functionality of the tool

4.1.3 Task List - PCAP Fabricator (Murat)

1. Relearn the prerequisite knowledge required for the application
    a. Python3 basics
    b. The layout of a data packet in a pcap file
2. Make a list of inputs that will allow customisation
    a. Making sure the inputs are features that are actually customizable in a pcap file
    b. Make sure the inputs are appropriate to the required context via sanitisation
3. Get an appropriate template of a pcap file in order to populate.
    a. Map the designated areas to append my inputs in order to save the users input to the pcap file
4. Save the file in a pcap format with the name that was provided by the user

4.1.4 Task List - Packer Analyser (Henry)

1. Research the required header files to help packet processing.
2. Make a suitable input method to take in .pcaps.

    2.1 Add sanitisation to check that a file was specified and that it is a .pcap.
3. Extract information from packets for processing e.g. protocol.
    3.1 Find further information to extract, Ip's, Ports, TTL.
4. Make a suitable, easy to read output method.
    4.1 Sort the list of outputs to see what ports/ips are used the most.
    4.2 Create statistics for graphing. (incomplete)

4.1.5 Task List - Steganography (Jack)

1. Research Steganography methods and how its performed
2. Research libraries that allow for image manipulation
3. Implement Text and image objects
    a. Provide sanitisation for image size and mode to check
4. Create the bitstream of the text as well as the delimiter
5. Replace the LSB value of each pixel with the encoded message
6. Save the output file
7. Implement the Steganography program to the front-end

4.1.6 Contribution list

Harry:
- Dos tool
- Documentation (1-2 & individual sections)

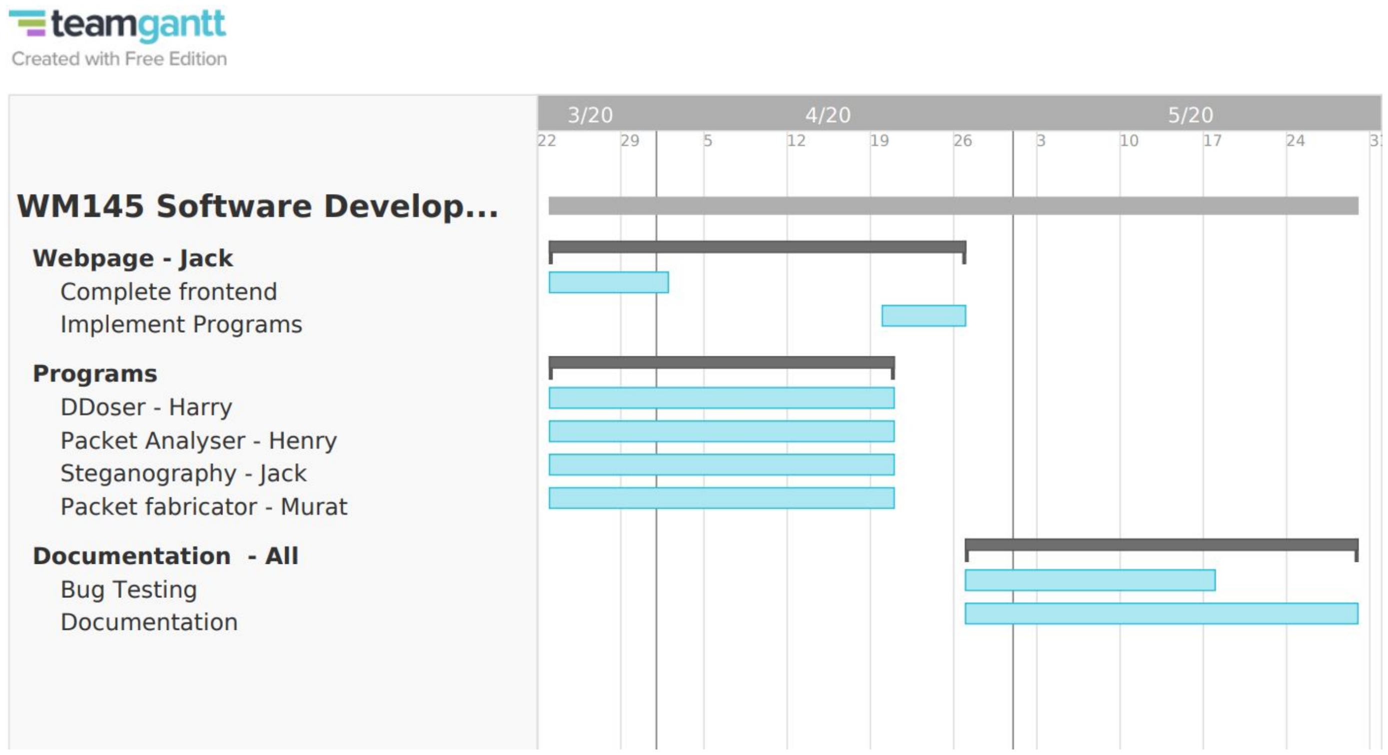
Henry:
- Pcap analyser tool
- Documentation (4-5 & individual sections)

Jack:
- Frontend web page
- Implementing tools to work with webpage
- Steganography tool
- Documentation (individual sections)

Murat:
- Pcap fabricator tool
- Documentation (3 & individual sections)

4.2 Time Dependencies

Refer to the GANTT graph to see task timeline.
Complete frontend - 4.1.1.2
Implement Programs - 4.1.1.2a
DDoser - 4.1.2
Packet Analyser - 4.1.4
Steganography - 4.1.5
Packet Fabricator - 4.1.3
Bug Testing - 4.1.1.3
Documentation - 4.1.1.4



4.3 Changelog

Requirement '4.1.1.2a' was extended from 7 days to 14 days due to misjudging how long the task would take. This pushed bug testing back by 7 days but otherwise, nothing else was affected.
Jacks toolkit changed from an SQL injector to a steganography toolkit. This is due to time constraints, this was decided during program research so no other tasks were affected.

5.0 Conclusion

In conclusion, the project went smoothly, the 2-week extension given to us due to COVID-19 allowed us to allocate more time to certain tasks which meant that all tasks were completed on time.

Weekly remote meetings were arranged to keep the group up to date on changes and activities related to the program. These meetings allowed us to ensure that the current progress was on track to be finished by the deadline, and if not we would be able to extend deadlines before it became an issue.

As a toolkit is near infinitely expandable, there is considerable room for improvement. Sections 3.2.1, 3.2.2, 3.2.3, 3.2.4, 3.2.5 detail potential upgrades to the individual tools.

Bibliography

Binarytides.com. 2012. *SYN Flood DOS Attack with C Source Code (Linux).*
Available at: https://www.binarytides.com/syn-flood-dos-attack/
[Accessed 30/05/2020].

inet_addr(3) - Linux man page. n.d.
Available at: https://linux.die.net/man/3/inet_addr
[Accessed 30/05/2020]

C IP Address Validation. n.d.
Available at: https://www.stev.org/post/cipaddressvalidation
[Accessed 30/05/2020]

NanoDano. (2015) Using libpcap in C [online] Available from:
https://www.devdungeon.com/content/using-libpcap-c#packet-type (accessed - 27/03/20)

Geeksforgeeks. (2017) Quicksort [online] Available from:
https://www.geeksforgeeks.org/quick-sort (accessed 09/04/20)

Github.com 2018 *steg/encodeimage.py at master xandhiller/steg Github.*
Available at: https://github.com/xandhiller/steg/blob/master/encodeImage.py
[Accessed 14/04/2020]

Youtube.com 2014. *Steganography Tutorial - Hiding Text inside an Image*.
Available at: https://www.youtube.com/watch?v=q3eOOMx5qoo
[Accessed 14/04/2020]

Youtube.com 2019 *Learning Flask (Playlist).*
Available at:
https://www.youtube.com/watch?v=BUmUV8YOzgM&list=PLF2JzgCW6-YY_TZCmBrbOpgx5pSNBD0_L
[Accessed 14/03/2020]

Github.com 2018. *steg/steg_img.py at master beatsbears/steg Github*
Available at: https://github.com/beatsbears/steg/blob/master/steg/steg_img.py
[Accessed 14/04/2020]

Miguelgrinberg.com 2017 *The Flask Mega-Tutorial*
Available at:
https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world
[Accessed 12/03/2020]

Stackoverflow.com 2014 *html Flask python button*
Available at: https://stackoverflow.com/questions/19794695/flask-python-buttons/19794878
[Accessed 13/05/2020]

Bootstrap.com n.d *navbar*
Available at: https://getbootstrap.com/docs/4.1/components/navbar/
[Accessed 16/03/2020]