SHRI VILEPARLE KELAVANI MANDAL'S

SHRI BHAGUBHAI MAFATLAL POLYTECHNIC

S Y N O P S I S

on

'Packetenizer'

Diploma

in

Computer Engineering

Submitted by:

1. **Brijesh Ghonia (1881011)**
2. **Kunal Joshi (1881018)**
3. **Jainam Joshi (1881019)**

Under the guidance of:

Mr. Janardan Kulkarni

Department of Computer Engineering

(2020 - 2021)

Contents

Chapter 1

Abstract

Continuous network monitoring using tools like Wireshark, TCPDump has become the norm of most enterprise networks. Interpreting the dump files produced by these tools requires some level of technical expertise and as well as time, one must scroll through seemingly countless numbers of packets to be able to make sense of it. While tools like Wireshark do provide some level of interpretation it still lacks the ability to extract solid data points from such dump files. In this project we aim to solve this problem by introducing a module that can interpret these dump files and extract solid data points. The output produced by the module would be interpretable even by non-technical people.

Chapter 2

Problem Statement

Business today is driven by data and powered by networks, enterprises these days have large networks and as a result large amounts of data is being transferred. There are stacks that manage large networks and its resources, systems that can provide real time threat protection, advanced firewalls that can protect networks for ingenuine traffic, however, there is a need for a system that can provide the aftermath of network from dump files captured by real time monitoring tools. With the dump files there is always a level of technical expertise and generally a lot of time is spent trying to interpret the data. As a result of this overhead, often key data points are missed out that could be valuable to an enterprise's network. It is not feasible for a person to sit hours in front of a screen scrolling through an endless list of packets trying to make sense of it. It is not just about the time being spent but also it will lead to human errors; it is also not the fault of the person the very nature of this job is susceptible to errors even for the best of us. The module being developed in this project will help in extracting key data points.

Chapter 3

System Requirements

The core module of the system will be written in Python 3.9 and thus a suitable environment along with various modules including but not limited to Scapy. A decently capable VPS (Virtual Private Server) with GNU/Linux Operating System.
The server side of this application will also be written in Python 3.9 using Flask micro framework. This server will not only host the front end of the application but will also serve the core module by exposing API endpoints. Since both core module and server side rely on Python 3.9, they can share the environment on the VPS. The VPS must come installed with a Web Server/Proxy (Nginx/Apache) to be able to serve the application.
The front end of the application will be structured using HTML and programmed using JavaScript. The front end can be hosted from the backend itself and no dedicated server is necessary for it. Bootstrap as a CSS framework will be used to style various components.

Server Side:

- VPS in any major cloud provider
- GNU/Linux OS
- Python 3.9 environment
- Active internet connection to host the application
- Web Server/Proxy (Apache/Nginx)

Client Side:

- Any modern web browser with JavaScript support (E.g. Chromium 80, Firefox 80)
- Active internet connection to connect to the server

Chapter 4

Proposed System

The system is aimed to extract the key data points from the dump generated from the real time monitoring tools. Thus, the user is required to first locate the dump file from the host machine and upload the same onto our server via the web-based interface. The web-based interface will be a gateway to the server which will handle all the processing and is responsible for extracting the important information from the dump file.

The dump file received will then be parsed packet by packet. Each Packet will then be scrutinized to uncover all the hidden information. The generated information will then undergo various checks where each bit of information will be checked and will be compared to specific parameters to ensure the Quality of Service. If any malicious / faulty packet is encountered in this process, it will be flagged and statistically analysed to check if there is any repetition of pattern encountered. This can also help the software to identify whether there is any Flooding or DOS Attack.

The converted output will be then sent to the web-based interface for displaying it to the user. User will then come into acquaintance with all the malicious activities and thus can take the necessary steps to prevent further continuation of such activities.

The Web Interface

The Web based interface serves the purpose of providing an interface to the user through which he/she can interact with the software. It will provide users with an intuitive and materialistic user interface. Web based interfaces will deprive the host machine from any unnecessary processing overhead and also make the interface platform independent.

The Core Module

The Core Module is responsible for parsing packets and retrieval of information. The received file will be parsed by Scapy module. The parsed packets will be used to extract core data points. Data structures will be used to maintain the state of various connections. Packets will also be passed to a statistic model to determine whether an attack was made on the network. Such data structures will be represented as JSON and sent back to the client to be rendered.

The Server-Side Processing Module

The Server-Side Processing Module is responsible for managing user requests and handing them over to the appropriate modules. It serves up the front-end module and exposes the core module by the means of API endpoints.

Chapter 5

Estimation & Planning

The Gantt Chart below shows a complete timeline this software will undergo during the developmental phase.

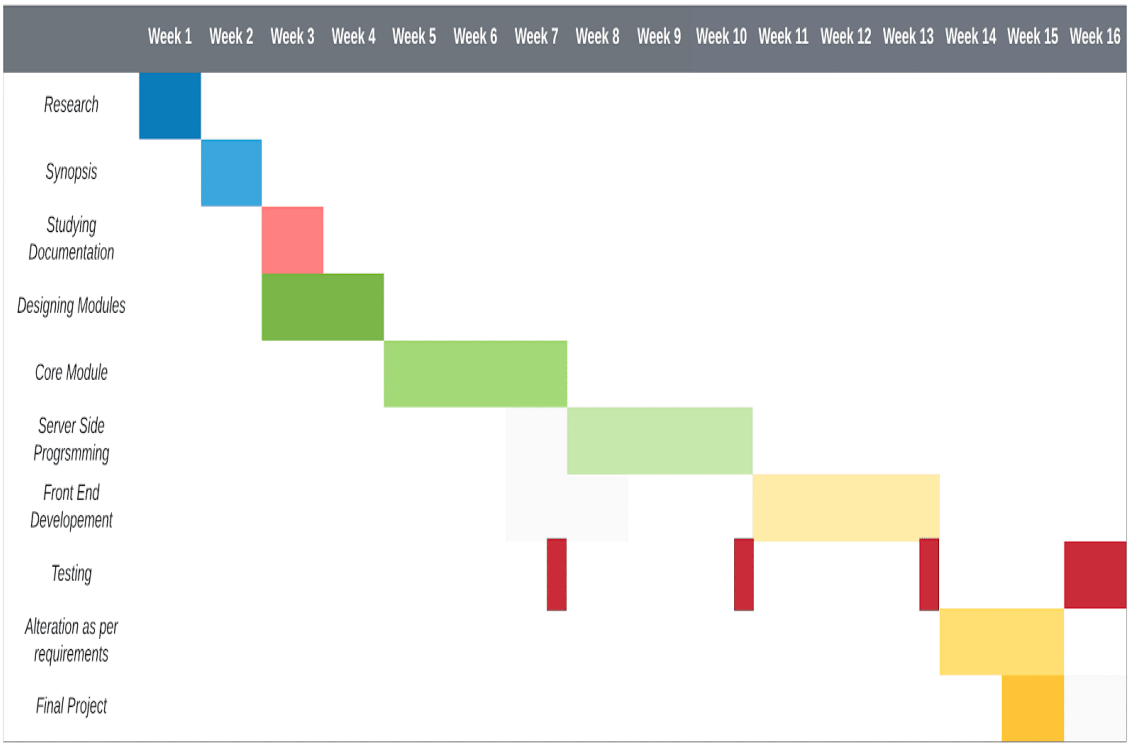

Figure 5.1: Gantt Chart for project development

- Starting up with the research first week will mark the completion of the research.
- Synopsis will be completed within the second week.

- In the third week initial preparation and the documentation of the required libraries will be studied.
- Parallelly the modules for the entire website will be designed. Its completion will mark the end of fourth week.
- The beginning of fifth week will initiate the coding phase. The Core Module will be completed in 3 weeks of time. At the end of seventh week the testing of Core Module will also be conducted hand in hand.
- Server-Side Programming will be started on the eighth week of the timeline. It also will be completed in 3 weeks of time along with its testing.
- Eleventh week will start by initiating the coding of Front End. Corresponding testing will be started by the end of thirteenth week, thus, completing the Front-End Development.
- Changes, if any, are advised, will be completed during fourteenth and fifteenth week.
- The final project will also be completed in the fifteenth week.
- The entire project will be tested to ensure an error free software in the sixteenth week.
- Thus, the entire software will be developed in 16 weeks of time.

Chapter 6

Diagrams

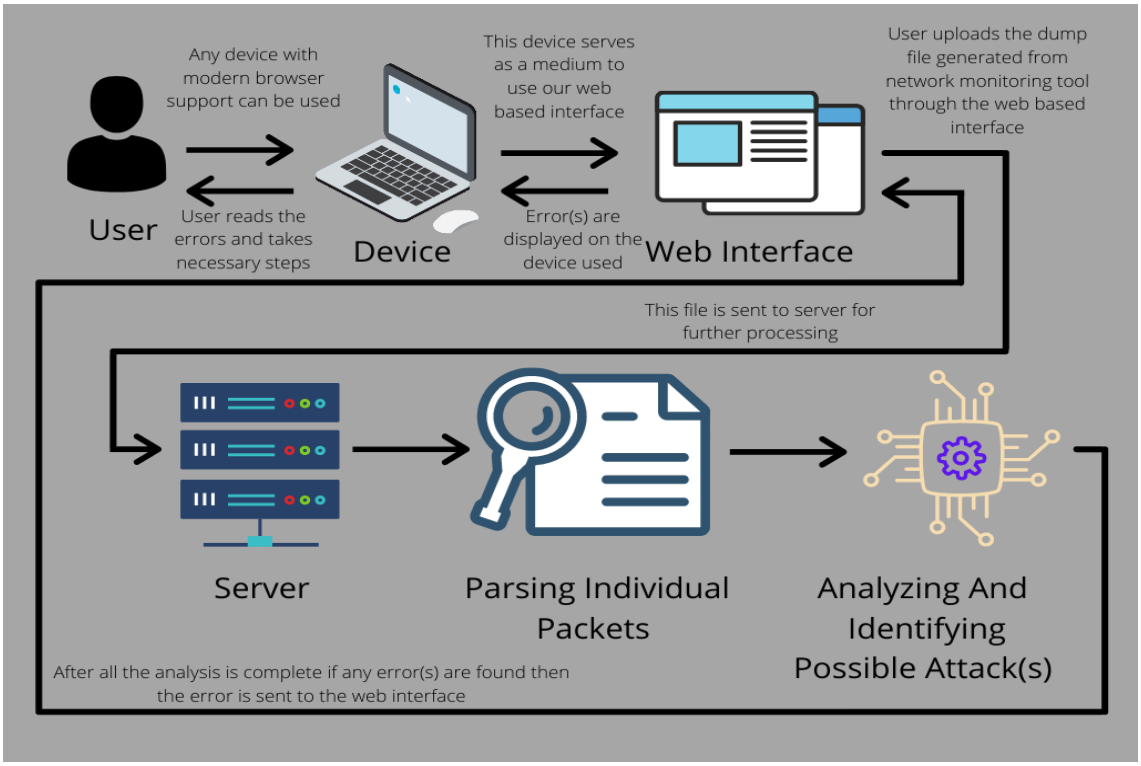

Figure 6.1: Block diagram of the project

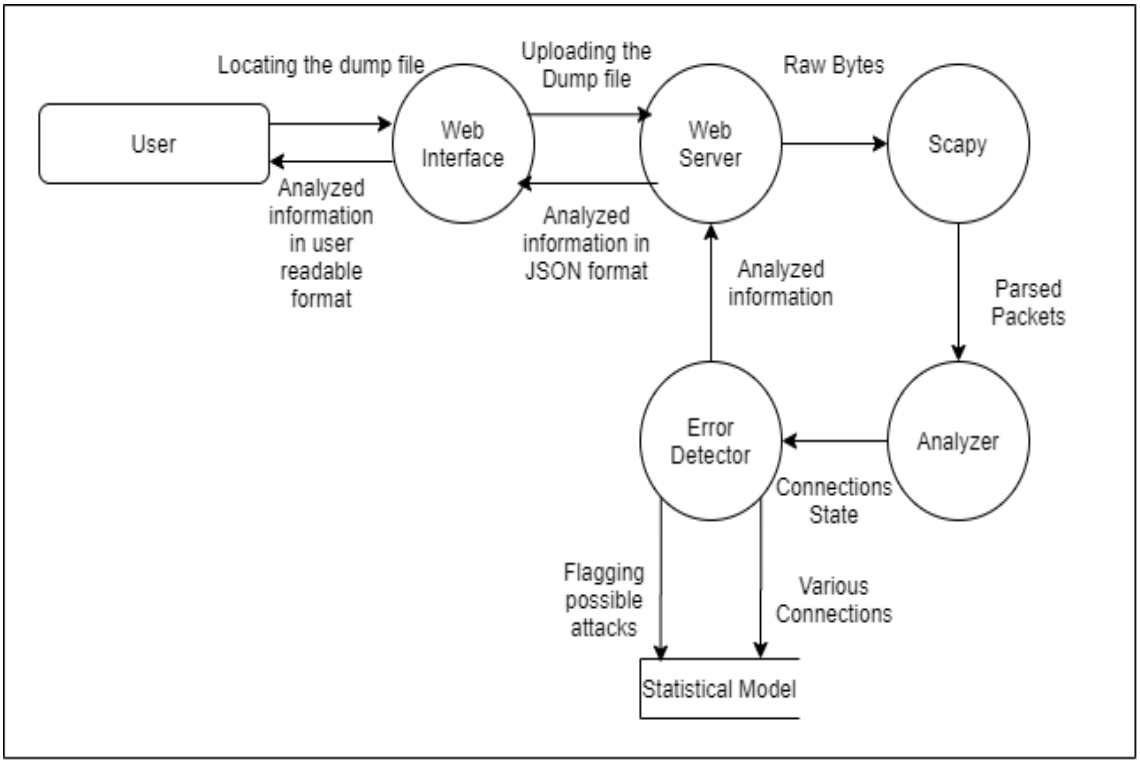

Figure 6.2: Data flow diagram of the project

Chapter 7

Future Scope

1. Support for various protocols can be added. E.g. Inclusion of more layer 3 protocols such as ATM, FDDI etc. Layer 4 protocols like IGMP, IPSec, RIP etc. Support for more VPN protocols.

2. Designing a better UI, inclusion of a framework to avoid code repetition seen when developing front end applications.

3. Developing a mobile app making it easy to analyze data on a fly.

4. Improve the statistical model to generate correct results and reduce the rate of false positives and true negatives when trying to detect various forms of attacks.

5. Support for user authentication and providing a more enhanced user tailored dashboard.

6. Designing and coding the modules to the standards to allow it to be accepted in the official Python modules list.

7. Releasing the project source code to the public on Github™ and resolving issues as they are notified and adding features as requested.

Chapter 8

Conclusion

Active network monitoring is amongst the most efficient network monitoring techniques. Most corporate organizations use Active Network Monitoring to ensure the security and integrity of their networks. This is usually backed by an employee whose job is to examine the output of the Active Monitoring Tool and identify any malicious activity or error. It also requires the employee to have a brief technical knowledge. Due to human intervention there are slight chances of human errors. Packet analysers can be used as an aid in these circumstances.

This project will analyse the network traffic and detect possible attacks like DOS/DDOS attempts, unintended connections to SSH, FTP and other services. This will help the user to identify and solve their network problems easily without the need of any technical knowledge. Also, the tedious and time-consuming job of manually examining the monitoring tool is eliminated. Thus, the resulting human resource can be utilized for other important tasks.

Chapter 9

References

1 Scapy Documentation

2 A similar application

3 Practical Packet Analysis by Chris Sanders

4 Methodologies for detecting DoS/DDoS attacks against network servers

5 Khundrakpam Johnson Singh(2017): Mathematical modelling of DDOS attack and detection using correlation

6 Analyzing TCP Dumps

7 TCP Dump guide