YarWeb: Web-based Generic YARA Rule Generator

A PROJECT REPORT

Submitted by,

Mr. SHREYAS BIJU	**20201CCS0108**
Mr. KAILAS M K	**20201CCS0118**
Mr. LAALAS TADAVARTHY	**20201CCS0119**
Mr. GOWRISHANKAR T O	**20201CCS0121**

Under the guidance of,

Ms. SRIDEVI S

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING (CYBER SECURITY)

At

PRESIDENCY UNIVERSITY

BENGALURU

JANUARY 2024

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE ENGINEERING

CERTIFICATE

This is to certify that the Project report **"YarWeb"** being submitted by "SHREYAS BIJU, KAILAS M K, LAALAS TADAVARTHY, GOWRISHANKAR T O" bearing roll number(s) "20201CCS0108, 20201CCS0118, 20201CCS0119, 20201CCS0121" in partial fulfilment of requirement for the award of degree of Bachelor of Technology in Computer Science and Engineering (Cyber Security) is a bonafide work carried out under my supervision.

Ms. SRIDEVI S
Assistant Professor
School of CSE&IS
Presidency University

Dr. S.P. Anandaraj
Professor & HoD
School of CSE&IS
Presidency University

Dr. C. KALAIARASAN
Associate Dean
School of CSE&IS
Presidency University

Dr. L. SHAKKEERA
Associate Dean
School of CSE&IS
Presidency University

Dr. SAMEERUDDIN KHAN
Dean
School of CSE&IS
Presidency University

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE ENGINEERING

DECLARATION

We hereby declare that the work, which is being presented in the project report entitled **YarWeb** in partial fulfilment for the award of Degree of **Bachelor of Technology** in **Computer Science and Engineering (Cyber Security)**, is a record of our own investigations carried under the guidance of **Ms. SRIDEVI S, ASSISTANT PROFESSOR, School of Computer Science Engineering, Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

Student Name(s)	Roll Number(s)	Signature(s)
Shreyas Biju	**20201CCS0108**	
Kailas M K	**20201CCS0118**	
Laalas Tadavarthy	**20201CCS0119**	
Gowrishankar T O	**20201CCS0121**	

ABSTRACT

In the modern digital landscape, the threat of malware and adware has significantly escalated, posing a challenge to the security of sensitive information. YarWeb, a web-based application, emerges as a critical tool in this context, revolutionizing the generation of YARA rules for effective malware detection. Designed with both novices and experts in mind, YarWeb simplifies the complex process of cybersecurity, making it accessible and user-friendly.

At its core, YarWeb excels in automating the creation of YARA rules from user-uploaded malicious files, streamlining the detection process with an impressive average accuracy of 0.80. This functionality not only accelerates the identification and mitigation of cybersecurity threats but also enhances user engagement in digital security practices.

Beyond its technical prowess, YarWeb serves as an educational platform, increasing awareness and understanding of cybersecurity risks among a broader audience. Its role in educating and empowering users is pivotal, contributing to the development of a more knowledgeable and resilient online community.

As a search engine-based model, YarWeb's versatility and practicality stand out, embodying the integration of advanced technology with a user-centric approach. It marks a significant advancement in cybersecurity tools, bridging the gap between sophisticated technologies and comprehensive digital security education, thereby positioning itself as an invaluable asset in the ongoing battle against cyber threats.

Keywords - Cybersecurity, YARA rules, Generic Rules, Automation Efficiency, Malware signatures, Malicious Strings.

ACKNOWLEDGEMENT

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Dean, School of Computer Science Engineering & Information Science, Presidency University for getting us permission to undergo the project.

We record our heartfelt gratitude to our beloved Associate Deans **Dr. Kalaiarasan C and Dr. Shakkeera L,** School of Computer Science Engineering & Information Science, Presidency University and **Dr. S.P. Anandaraj**. Head of the Department, School of Computer Science Engineering, Presidency University for rendering timely help for the successful completion of this project.

We are greatly indebted to our guide **Ms. Sridevi S, Assistant Professor**, School of Computer Science Engineering, Presidency University for their inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the project work.

We would like to convey our gratitude and heartfelt thanks to the University Project-II Coordinators **Dr. Sanjeev P Kaulgud, Dr. Mrutyunjaya MS** and also the department Project Coordinator **Ms. Manasa C M**.

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

Shreyas Biju

Kailas M K

Laalas Tadavarthy

Gowrishankar T O

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER-1
INTRODUCTION

1.1 Background

The cybersecurity landscape is witnessing an unprecedented evolution in malware complexity, making the battle against cyber threats increasingly challenging. Traditional malware analysis methods, primarily manual and prone to inaccuracies, struggle to keep pace with these sophisticated threats. In this context, YARA emerges as a potent tool with its rule-based detection approach, offering a high degree of precision. However, its heavy reliance on command-line interfaces poses significant hurdles for non-technical users, creating a gap in accessibility and ease of use. YarWeb steps in to bridge this gap, presenting a groundbreaking, user-friendly web-based platform. It enables efficient generation and analysis of YARA rules, thereby simplifying the complex process of malware detection and making it more accessible to a broader range of users, regardless of their technical proficiency.

1.2 Problem Statement

The command-line interface of YARA, while powerful, presents a steep learning curve for those without a background in programming. This technical complexity not only limits its user base but also hampers its potential for widespread adoption in diverse environments. Moreover, the dynamic and evolving nature of malware presents further challenges. The existing YARA rule sets, often plagued by redundancies, struggle to keep up, leading to inefficiencies, delays, and inaccuracies in threat detection. Addressing these issues is essential for maintaining robust and effective cybersecurity measures in a rapidly changing digital landscape.

1.3 Overview

YarWeb is more than just a tool, it's a paradigm shift in malware analysis. By simplifying YARA rule creation and analysis, it opens the door to a broader audience, including those with limited technical skills. Its intuitive interface is a game-changer, significantly reducing manual labor and enhancing detection speed and accuracy. This democratization of malware analysis marks a significant step forward, making sophisticated cybersecurity tasks more accessible and engaging for a diverse range of users. YarWeb's impact extends beyond just professional domains; it's a valuable asset for educational institutions and businesses, equipping them with the tools to effectively combat the ever-increasing complexity of malware threats. In essence, YarWeb is redefining the landscape of cybersecurity, making it more inclusive, efficient, and responsive to the challenges of modern digital threats.

CHAPTER-2
LITERATURE SURVEY

1. David Bernal Michelena. "Detecting Malicious Files with YARA Rules as They Traverse the Network using":

This paper discusses enhancing network security by using YARA, a tool for detecting suspicious files, in combination with Zeek IDS and a custom script to scan network traffic. The approach focuses on early detection of cyberattacks, with an emphasis on minimizing false positives in YARA rules to reduce unnecessary alerts. Future developments include adding more notification methods, sandbox integration for analysing suspicious files, and adapting to encrypted protocols. The paper also underscores the need to upgrade to Python 3 for improved handling of malware scenarios.

2. Michael Brenge, Christian Rossow." YARIX: Scalable YARA-based Malware Intelligence" CISPA Helmholtz Centre for Information Security:

The paper introduces a methodology using YARIX to efficiently identify files matching specific YARA rules, particularly useful in handling large malware datasets. This method optimizes malware pattern searching, improving both storage requirements and search performance. YARIX demonstrates scalability in storing and searching vast malware samples, addressing the limitations of traditional methods which can be slow or storage-intensive for large malware 7datasets analysis.

3. Nitin Naik, Paul Jenkins, Roger Cooke, Jonathan Gillett, and Yaochu Jin. "Evaluating Automatically Generated YARA Rules and Enhancing Their Effectiveness,":

This research explores ways to boost the effectiveness of automatically

generated YARA rules for malware analysis. It combines YARA's flexibility with fuzzy hashing to improve detection accuracy. However, challenges remain: attackers can manipulate or hide indicators, extracting those indicators requires expertise, and balancing their number in rules is crucial. While automation helps, these rules may need optimization and might not catch unique threats as well as manually crafted ones. Overall, the research showcases YARA's potential but also highlights existing limitations that need further attention.

4. Dominika Regeciova , Dusan Kolar , and Marek Milkovic. "Pattern Matching in YARA: Improved Aho-Corasick Algorithm":

The paper by Dominika Regeciova, Dusan Kolar, and Marek Milkovic focuses on improving the efficiency of the YARA tool, specifically in processing regular expressions. It addresses the limitations of the Aho-Corasick algorithm used in YARA, which affects scanning speed when using regular expressions for pattern matching in malware analysis. The authors propose an enhanced version of this algorithm, demonstrating through experiments its potential to significantly speed up pattern matching, thereby boosting the overall efficiency of YARA in detecting malware.

5. Nitin Naik, Paul Jenkins, Nick Savage, Longzhi Yang3, Kshirasagar Naik4 and Jingping Song "Augmented YARA Rules Fused with Fuzzy Hashing in Ransomware Triaging":

The paper addresses enhancing malware triaging, focusing on ransomware. It evaluates the effectiveness of various methods, including fuzzy hashing, import hashing, and YARA rules. The study finds that combining fuzzy hashing with YARA rules significantly improves triaging of ransomware and other malware types. This integrated approach offers a more effective tool for malware analysts in identifying and combating cyber threats.

6. "Generation of Static YARA-Signatures Using Genetic Algorithm" by Dominika Regeciova, Dusan Kolar, and Marek Milkovic:

The paper presents a novel approach for generating static YARA signatures for malware detection using a Genetic Algorithm (GA). It compares a GA-based method with Multinomial Naive Bayes and Maximization-Maximization algorithms, highlighting GA's innovative application in YARA rule generation. The GA method shows superior detection scores with malware datasets, but faces challenges like high computational complexity and lengthy training time. These limitations and the potential for future improvements in GA's efficiency against complex malware form the crux of the paper's research.

7. "Automatic YARA Rule Generation" by Myra Khalid, Maliha Ismail, Mureed Hussain, and Muhammad Hanif Durad:

The paper introduces an automated framework for creating YARA rule-based malware signatures, focusing on handling the surge of new malware. It employs a modular approach with various units for parsing, scoring, filtering, composing, and evaluating malware, using a pre-trained Naïve Bayes model for efficiency. The framework boasts an average precision of 0.95 in malware detection. However, its effectiveness may depend on the quality of the training data for the Naïve Bayes model, which could impact the accuracy of the generated rules.

8. "Fuzzy Hashing Aided Enhanced YARA Rules" by Nitin Naik, Paul Jenkins, Nick Savage, Longzhi Yang, Kshirasagar Naik, Jingping Song, Tossapon Boongoen, and Natthakan Iam-On:

This paper addresses the challenge of effectively coping with the vast amount of malware created daily. It focuses on enhancing malware triaging, which involves

separating likely from unlikely malware samples. The paper specifically looks at improving the detection rate of YARA rules, commonly used for malware detection and classification based on string and pattern matching. While YARA rules are effective, they can become bulky and complex, affecting performance. The proposed solution integrates fuzzy hashing with YARA rules to improve detection rates without significantly increasing complexity and overheads. This approach employs fuzzy hashing, import hashing, and YARA rules for triaging, and through extensive experiments, it demonstrates that the enhanced YARA rules achieve better detection rates compared to traditional methods. However, a potential disadvantage is the added complexity and the need for fine-tuning the integration of fuzzy hashing with YARA rules to ensure optimal performance and accuracy in diverse malware scenarios.

CHAPTER-3

RESEARCH GAPS OF EXISTING METHODS

3.1 Existing Systems

Several existing systems and tools are employed for YARA rule generation, searching, and management:

- Avast YaraNG: A revamped YARA scanner designed for large-scale operations, featuring efficient handling of extensive rulesets, HyperScan integration for advanced pattern matching, and the ability to transpile YARA rules into C++ for optimized processing.
- VirusTotal's YARA: A versatile, multi-platform tool for malware classification, compatible with various operating systems and accessible via command-line or Python scripts. It is augmented with extensions like YARA-CI for continuous rule testing and extend for scanning compressed files.
- Sophos YaraML: Developed by SophosAI, this tool leverages machine learning to generate YARA rules, analysing datasets of benign and malicious strings. It's user-friendly for both novices and experts in machine learning, making it a practical choice for incident responders and malware researchers in diverse scenarios.

3.2 Drawbacks of Existing Systems

- Here are some drawbacks of using YARA for malware detection, including the specific limitations identified for the YARA-based systems:
- Signature Evasion: Attackers can bypass YARA's signature-based detection by using crypting services, packers, and polymorphism, creating malware variants that don't match existing signatures.

- Delayed Detection: In scheduled scans, malware may not be discovered until a significant amount of time after it is introduced to a system.
- Performance Impact: Real-time scanning with YARA rules can affect system performance and might cause critical processes to crash.
- Risk of Disabling: Threat actors may discover and disable the scheduled scans or delete the rulesets.

CHAPTER-4

PROPOSED METHODOLOGY

4.1 Proposed work

YarWeb, as a pioneering web-based YARA search engine, represents a significant advancement in cybersecurity, particularly in malware detection. This project stands out for its detailed attention to both functional and non-functional requirements, aiming to deliver an exceptional user experience, robust security measures, and high-performance capabilities. The development process of YarWeb has been meticulously planned and executed, focusing on creating a platform that is not only powerful in its technical capabilities but also intuitive and user-friendly. This approach ensures that YarWeb can be effectively used by a wide range of users, from cybersecurity professionals to those with limited technical knowledge.

4.2 Features of Proposed Work

The project introduces several innovative features:

- User Experience: The interface of YarWeb is crafted to be intuitive and straightforward, making complex malware detection processes accessible to all users.
- Malware Detection Efficiency: The platform utilizes cutting-edge algorithms for swift and accurate identification of various malware forms thereby enhancing the efficiency of the detection process.
- Immediate Feedback: YarWeb provides instant feedback to users, making the platform interactive and more effective in real-time analysis.
- Accessibility and Usability: With a focus on broad accessibility, YarWeb is designed to cater to a diverse user base, simplifying complex

cybersecurity tasks without compromising on the depth of its functionalities. Sacrificing technical capabilities.

4.3 Advantages of Proposed Work

- Enhanced Cybersecurity: The platform significantly accelerates the process of detecting and mitigating cyber threats, thereby reinforcing digital security measures across various domains.
- Educational and Empowering: YarWeb serves as an educational tool, enhancing understanding and proficiency in YARA rule creation and analysis, thereby empowering users across different levels of expertise.
- Rule Performance Optimization: A distinctive aspect of YarWeb is its ability to optimize YARA rule performance. By refining rule management and eliminating redundant elements, the platform ensures more accurate and efficient malware detection, which is essential in countering sophisticated and rapidly evolving cyber threats. The backend infrastructure of YarWeb is particularly noteworthy, as it manages the complexities of rule generation and maintenance, further enhancing the platform's reliability and efficiency.

CHAPTER-5
OBJECTIVES

1. **Advanced Malware Detection Enhancement:** This project aspires to elevate the effectiveness of YARA rules for detecting a broad spectrum of malware. This enhancement is vital for quick identification and neutralization of cyber threats, thereby strengthening cybersecurity defenses across various platforms.

2. **Streamlining Rule Generation through Automation:** A central aim is to automate the generation of YARA rules, thereby reducing manual efforts and ensuring a swift, adaptive response to new and evolving malware threats.

3. **Refined Signature Selection Mechanism:** Incorporating a sophisticated search engine, the project aims to enable precise selection of the most effective YARA signatures, tailored for specific malware types. This mechanism is designed to increase the efficiency and accuracy of the malware detection process.

4. **Enhanced Scalability for Large-scale Data Processing:** The project is structured to adeptly handle extensive datasets, crucial in an era where data volume is continuously expanding, ensuring consistent performance and reliability.

5. **Generality in YARA Rule Application:** The project emphasizes the development of YARA rules with wide applicability, thereby optimizing

resource usage and streamlining the process of managing and applying rules in diverse scenarios.

6. **Focused Improvement on Detection Accuracy:** A significant portion of the project is dedicated to refining the accuracy of YARA rules, aiming to substantially reduce the occurrences of false positives and negatives for more reliable malware detection**.**

7. **Designing an Inclusive User Interface:** The project envisages creating an intuitive and user-friendly interface, making advanced cybersecurity tools accessible and usable by a broader demographic, including those with minimal technical background.

8. **Expanding the Knowledge Base and Documentation:** The project plans to contribute extensively to the field of YARA rule creation, providing thorough documentation and practical examples, thereby enhancing the overall understanding and knowledge in this area.

9. **Cultivating a Collaborative Community and Feedback Mechanism:** An integral goal is to build an engaged community focused on YARA rule development and malware analysis. This community will be instrumental in fostering collaboration, sharing insights, and providing constructive feedback, thereby continuously refining and enhancing the tool's capabilities and applications.

CHAPTER-6

SYSTEM DESIGN & IMPLEMENTATION

6.1 Requirement Analysis

6.1.1 Hardware Requirements

To build and deploy the YAR WEB the following hardware requirements are recommended:

Minimum Requirements

- Intel Core i3 8th generation or later. AMD Ryzen 3 or later. (1.8 GHz minimum)
- 4 GB or RAM Minimum, 8GB recommended.
- 16 GB of hard Disk space. 64-bit

6.1.2 Software Requirements

1. Operating System:

- Windows, Linux.
- Cross-platform compatibility for the front end (e.g., web browsers).

2. Backend Technologies:

- Programming Language: Python for backend development.
- Pandas: A data manipulation and analysis library in Python, used for processing CSV data.
- Web Framework: Flask for API development. A lightweight WSGI web application framework in Python is used to build the web server.
- YARA Integration: Integration with the YARA tool for rule testing and validation.
- YARA-Python: Python library for YARA, used for writing and compiling YARA rules.

3. Technologies:

- HTML, CSS, and JavaScript for front-end development.
- Frameworks and Libraries: Flask, Pandas, Flask_SQLAlchemy, bcrypt, OS, YARA, secrets, requests, time, datetime

4. Database Requirements:

- The database system should be properly configured and optimized for performance and security.

- Data modeling to represent YARA rules, malware samples, metadata, and user data.

5. Development Tools:

- Integrated Development Environment (IDE) like PyCharm or Visual Studio Code for code development.
- Version Control System (e.g., Git) for source code management.
- Continuous Integration and Continuous Deployment (CI/CD) tools for automated testing and deployment.

6.2 Architecture Diagram

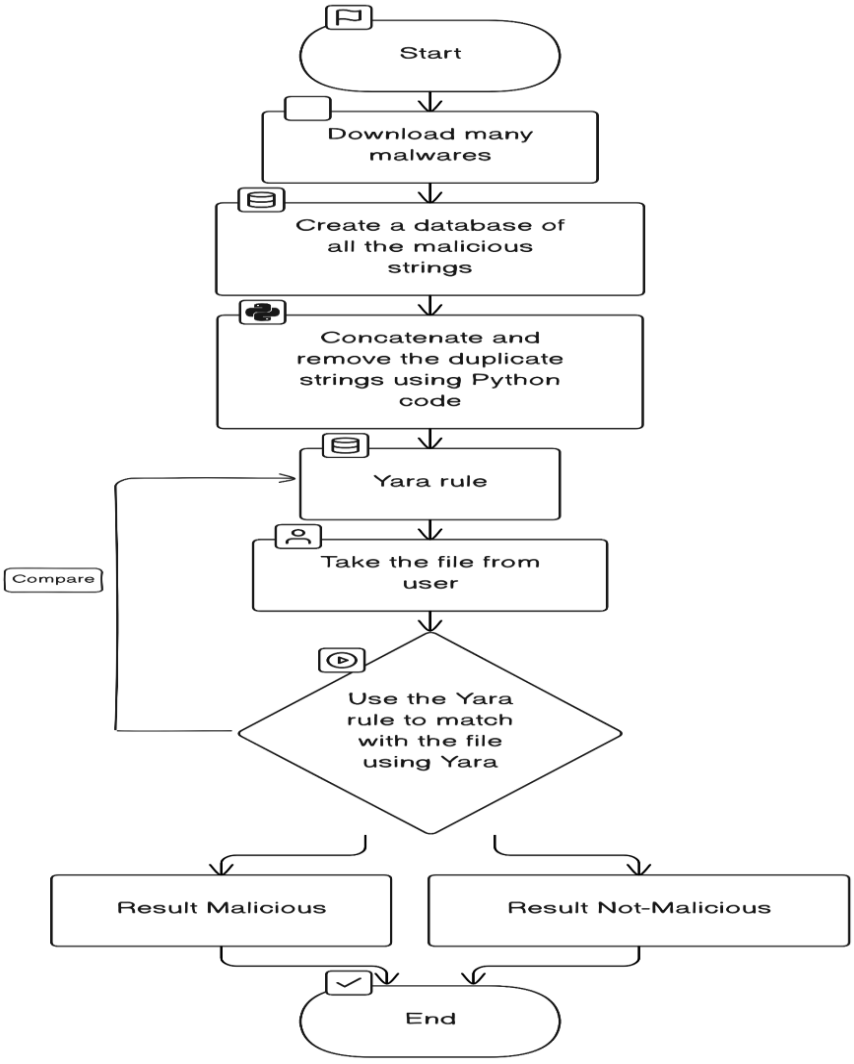

Figure 6.2.1 Architecture Diagram

6.3 Modules

The entire modules can be divided into 6 phases:

1. Database Creation: This step is about building a comprehensive database of malware signatures. It involves meticulous collection and organization of data, ensuring a rich and well-structured repository that forms the backbone for YARA rule generation.
2. Database Modification: The database undergoes a process of refinement and optimization, aimed at enhancing its compatibility with the YARA rule format. This step ensures that the data is clean, relevant, and formatted appropriately for efficient rule creation.
3. Generic YARA Rule Production: This module centers around a user-friendly web interface that facilitates the selection of malware signatures and the generation of corresponding YARA rules. The backend logic is designed for efficiency and accuracy, ensuring the rules are robust and reliable.
4. User Input Validation: This phase is critical for maintaining the system's security, involving stringent validation of user inputs, particularly for file uploads. It ensures the integrity of the system and protects against potential security vulnerabilities.
5. Testing YARA Rules: Users are provided with functionalities to test the effectiveness of the YARA rules. This testing is comprehensive, simulating real-world conditions to ensure the rules are effective in identifying malware.
6. Admin Monitoring and User Management: The final module involves creating an admin interface for monitoring user activities and managing the platform. This phase is crucial for understanding user behavior, system

usage, and for making informed decisions about future enhancements and updates.

6.4 Flow Charts

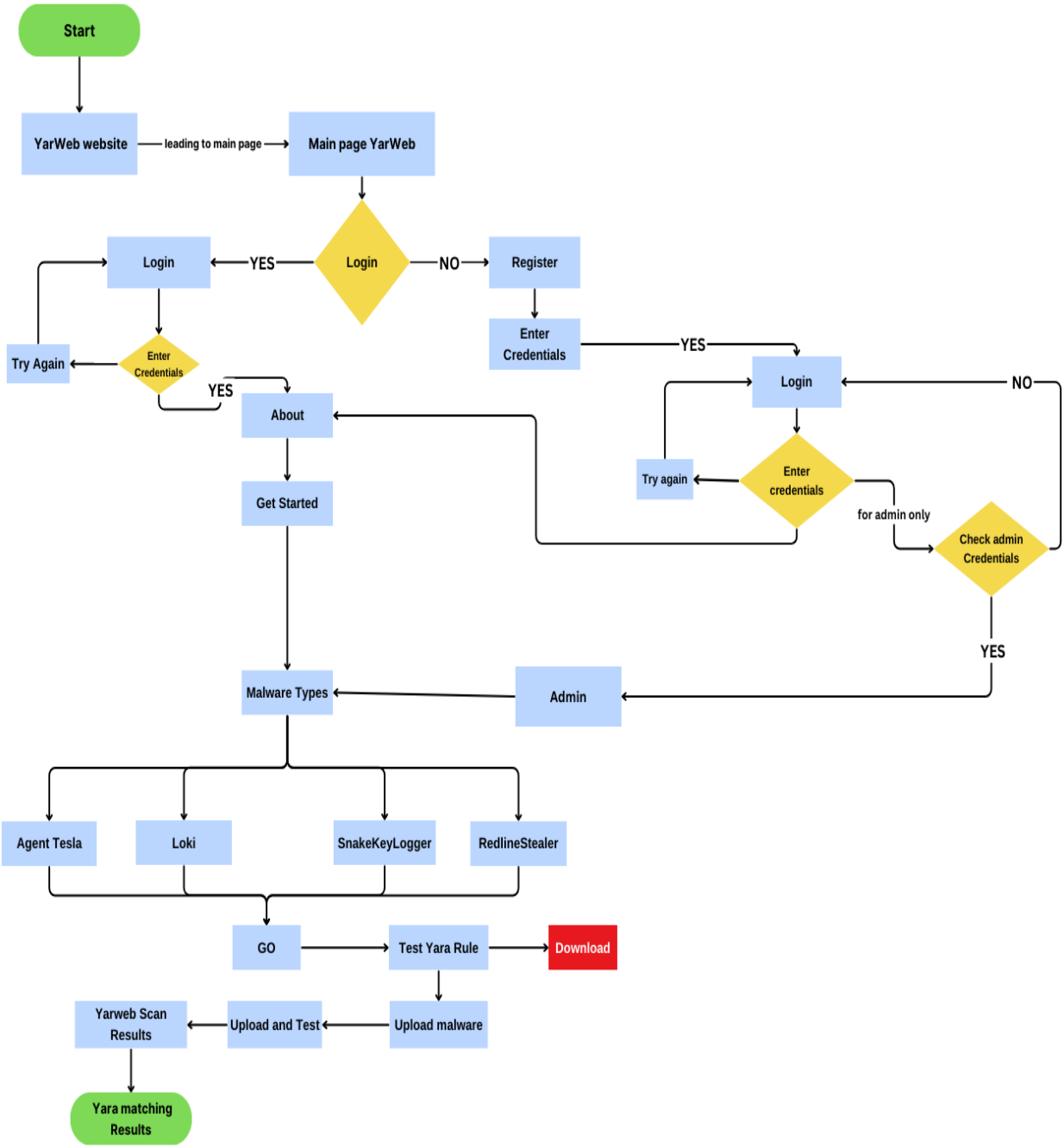

Fig 6.4.1 Overview

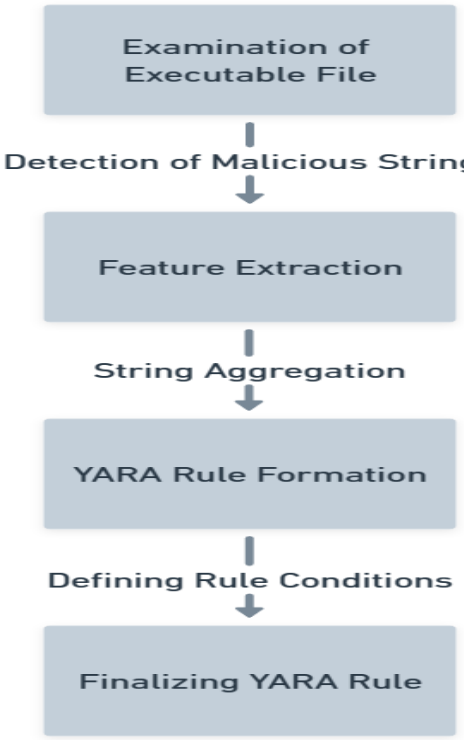

Fig 6.4.2 Yara rule formation

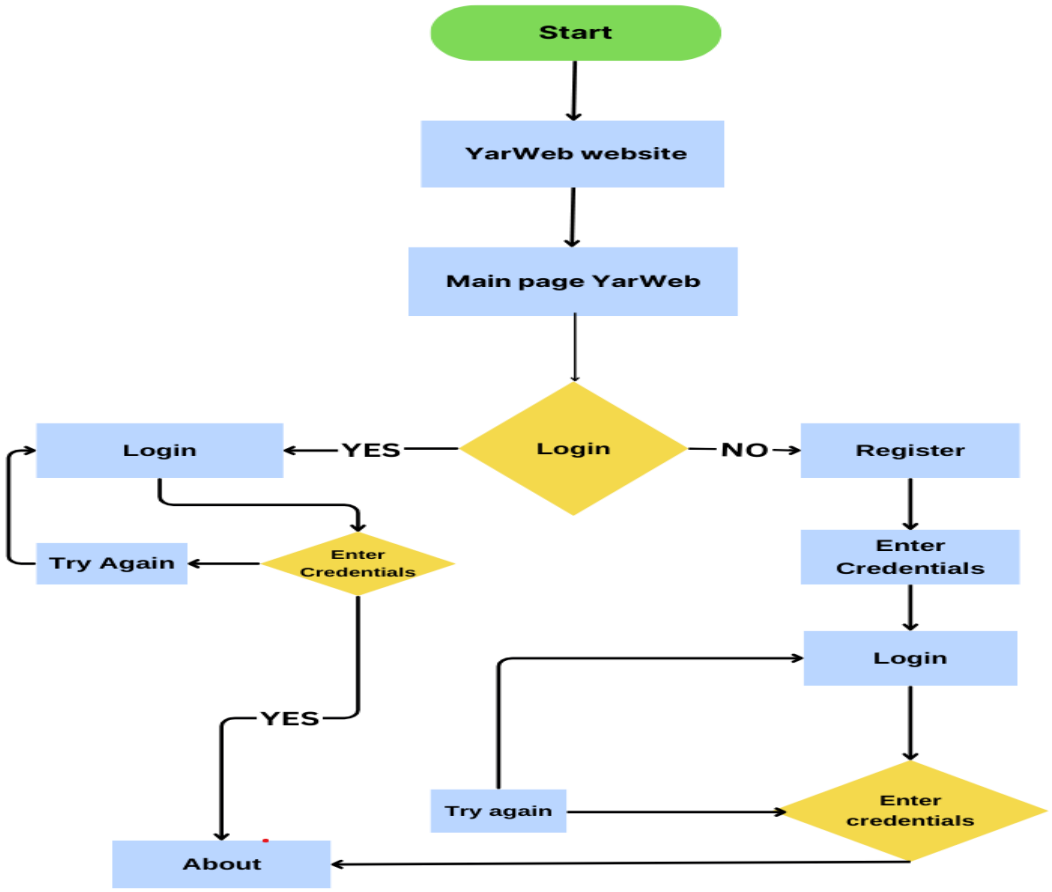

Fig 6.4.3 User login and workflow

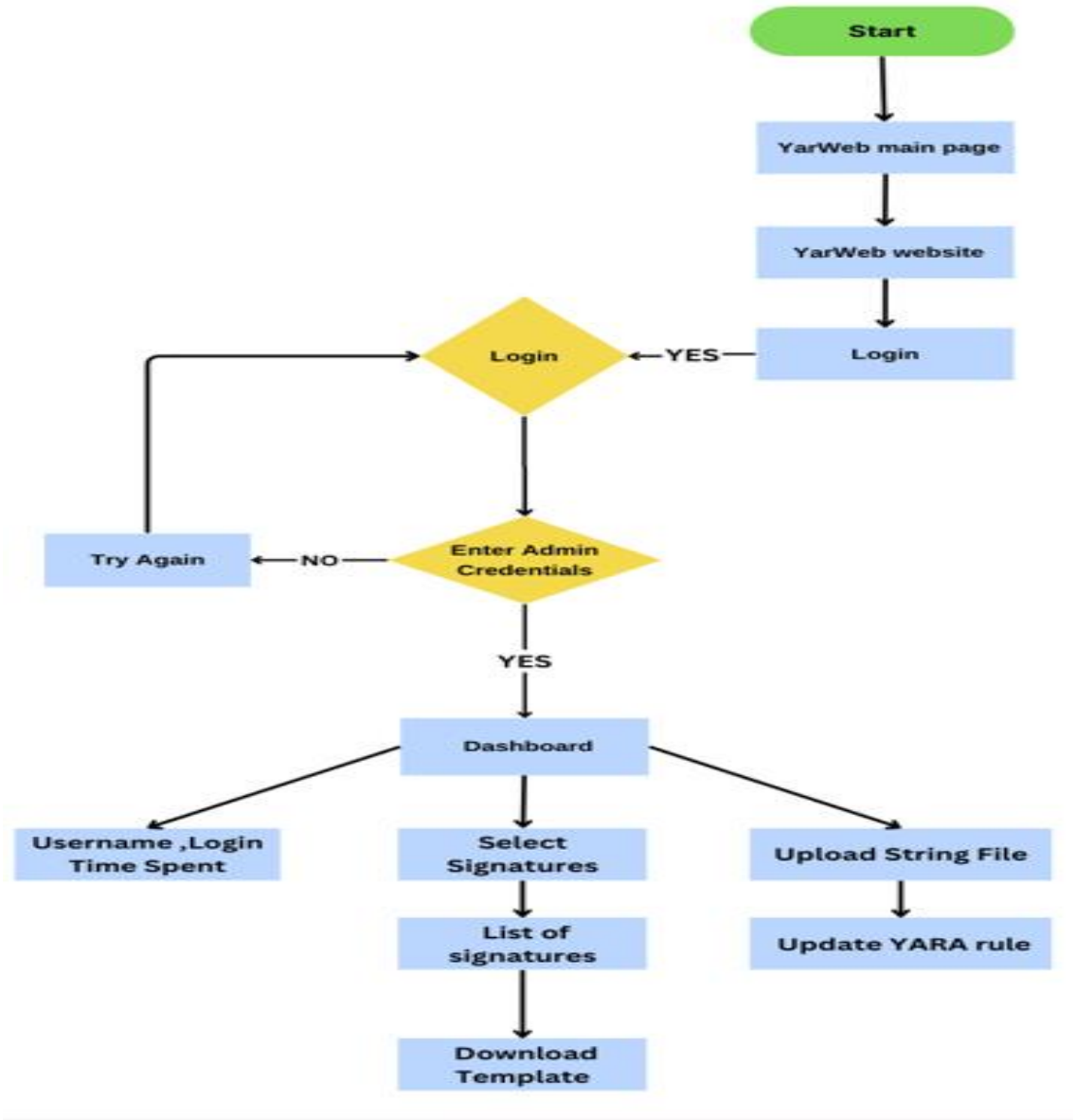

Fig 6.4.4 Admin's login and workflow

6.5 Testing

6.5.1 Frontend Testing

This includes verifying the form validation mechanisms that check the format and size of the user-uploaded files, ensuring they adhere to the specified criteria. The application's interface, where users interact to upload files and generate YARA rules, must be tested for usability, responsiveness, and compatibility across different browsers and devices. The feedback mechanisms that provide users with error messages or confirmations are also a critical part of frontend testing, ensuring that the application communicates effectively with the users.

6.5.2 Backend Testing

It deals with the server-side functionalities of the application, particularly those that process the uploaded files. This includes testing the server's response to the file uploads, the integration with the YARA-python library for rule testing, and the overall data handling and processing logic. Structural testing ensures that the database correctly stores and manages the data related to malware signatures and YARA rules. Functional testing checks the data flow and processing logic when users upload files and generate rules. Non-functional testing is also crucial, focusing on the performance of the backend under load (especially important given the potential volume of file uploads and rule processing) and the security aspects to protect against any data breaches or unauthorized access.

6.5.3 Deployment

Deployment refers to the process of setting up and launching the project, which integrates sophisticated front-end and back-end technologies to create a comprehensive and functional system. For the front-end, HTML, CSS, and JavaScript are utilized, complemented by frameworks such as React, Angular, or Vue.js to craft a dynamic and intuitive user interface. On the back-end, Python is employed for data processing, with Flask serving as the web framework for API development. This setup includes integration with the YARA tool, essential for malware rule testing and validation, and utilizes the YARA-Python library.

The project's database is meticulously configured to ensure high performance and security, correctly storing and managing YARA rules, malware samples, and relevant data. The

deployment process adheres to current best practices in software development, incorporating Continuous Integration and Continuous Deployment (CI/CD) for streamlined testing and deployment. Development tools like PyCharm or Visual Studio Code are used for coding, while Git is employed for effective source code management.

6.6 Maintenance

After deployment, the maintenance of the project involves a continuous process of enhancement and updates, especially focused on the website and graphical user interface (GUI). This maintenance strategy is essential to ensure improved performance and an optimal user experience over time. Regular updates and refinements are made to the website and GUI, responding to user feedback and evolving requirements. This includes the integration of additional databases for different malware signatures, which allows the tool to remain relevant and effective against an ever-changing landscape of cyber threats:

1. Website/GUI Refinement and Modification:
    - User Feedback Integration: Regularly incorporating feedback from users to improve the interface and functionality of the website/GUI.
    - Performance Optimization: Continuously updating the website and GUI for faster load times, better responsiveness, and smoother user interactions.
    - Design Updates: Implementing design changes to enhance user experience, including updating layouts, colors, and navigation elements to keep the interface modern and user-friendly.
2. Incorporation of Additional Databases for Various Malware Signatures:
    - Database Expansion: Continuously adding new malware signature databases to increase the comprehensiveness and effectiveness of the tool.

- Data Accuracy and Relevance: Regularly updating existing databases to ensure that the information remains accurate and relevant to current malware threats.
- Integration Testing: Ensuring that new databases are seamlessly integrated with existing systems without disrupting the tool's functionality.

3. Efficient YARA Rule Generation:
    - Automation Enhancements: Updating and refining the automation processes to make the generation of YARA rules even more efficient and less time-consuming.
    - Rule Set Updates: Regularly adding new rule sets and updating existing ones to keep up with the latest malware trends and threats.
    - User Experience in Rule Generation: Improving the interface and process for rule generation based on user feedback and usability studies to make it more intuitive and user-friendly.

CHAPTER-7

TIMELINE FOR EXECUTION OF PROJECT

(GANTT CHART)

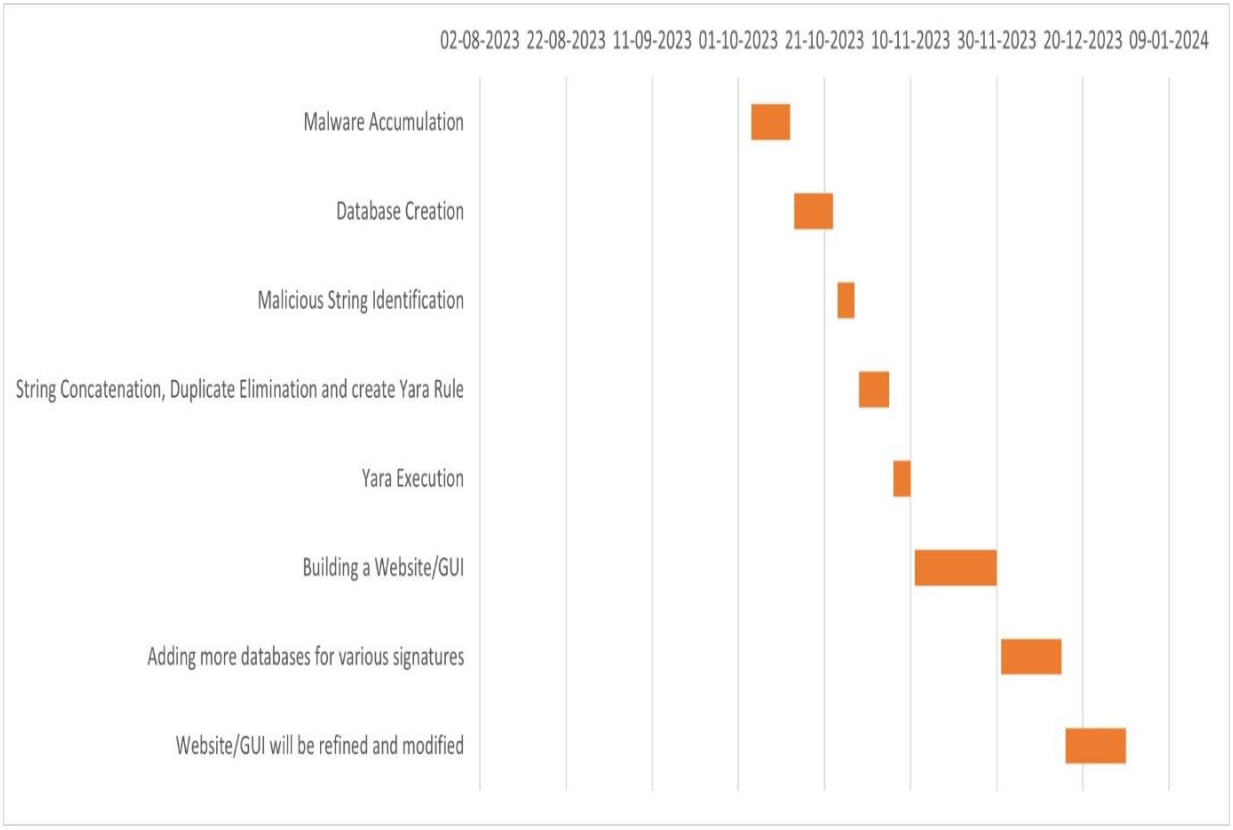

Fig 7.1 Gantt chart

CHAPTER-8
OUTCOMES

1. Efficient YARA Rule Generation:

Users can quickly generate YARA rules based on a selection of predefined malware signatures.

The process reduces the time and complexity typically associated with manually writing YARA rules.

2. Enhanced Malware Analysis Capabilities:

The tool provides cybersecurity professionals and researchers with a user-friendly platform to create and test YARA rules, aiding in the detection and analysis of malware.

The ability to test rules against .exe files enables immediate validation and refinement of the rules, thereby improving the effectiveness of malware detection efforts.

3. Improved Accuracy and Reliability:

By automating the rule generation process and providing a structured testing mechanism, the likelihood of human error is reduced.

The project aims to ensure that the generated YARA rules are syntactically correct and effective in identifying specified malware patterns.

4. User Empowerment:

The application empowers users with varying levels of expertise in YARA and malware analysis to efficiently create and test rules.

It also serves as an educational tool for those learning about malware signatures and pattern detection.

5. Scalability and Extensibility:

The project's design allows for easy updating and expansion of the malware signature database, ensuring the tool remains relevant and effective against evolving malware threats.

The modular design of the web application permits further enhancements and integration of additional features, such as more complex rule generation or integration with other malware analysis tools.

6. Community Contribution:

By providing an open and accessible platform for YARA rule generation and testing, the project contributes to the broader cybersecurity community.

It facilitates collaborative efforts in malware analysis and threat intelligence sharing.

CHAPTER-9
RESULTS AND DISCUSSIONS

9.1 Results

The YarWeb project has significantly advanced the field of cybersecurity with its innovative approach to YARA rule generation. By simplifying and automating the process of rule creation, the project addresses a critical need in cyber defense, markedly reducing the time and effort traditionally required in manual rule crafting. This advancement is particularly crucial in the rapidly evolving landscape of cyber threats, where the ability to swiftly respond and adapt is vital.

The tool's impact extends to enhancing malware analysis capabilities. It has become an invaluable resource for cybersecurity professionals, offering a powerful yet accessible platform for the creation and testing of YARA rules. The user-friendly interface of the tool is specifically designed to facilitate proactive identification and analysis of malware, enabling immediate validation and refinement of rules. This feature is essential in an environment where the nature and complexity of cyber threats are constantly changing.

A key aspect of the project is the integration of automation, which significantly bolsters the accuracy and reliability of the generated rules. Automated processes minimize human error and ensure a high level of precision in rule generation. Furthermore, the structured testing mechanism in place rigorously evaluates each rule for syntactical correctness and effectiveness, providing reliable tools for malware identification.

Beyond its functional capabilities, the YarWeb project contributes to the broader

cybersecurity community. Its scalable and extensible architecture ensures its relevance and adaptability to new malware challenges. The design supports the incorporation of updates and new functionalities, maintaining its effectiveness against evolving threats. Additionally, by offering an open platform for YARA rule generation and testing, the project promotes collaboration and intelligence sharing within the cybersecurity community, fostering a collective approach to combating cyber threats.

9.2 Future Enhancements

The development of YarWeb heralds a significant leap forward in the domain of digital security tools, with its focus on modularity, customization, and enhanced language support. This new iteration is being crafted in Rust, a language renowned for its security features and performance, positioning YarWeb to meet modern cybersecurity challenges more effectively. Such advancements in the core architecture promise to make YARA more adaptable to the diverse needs and emerging threats faced by users, ensuring that it remains at the forefront of malware detection and analysis.

Concurrently, the evolution of user interfaces and editing tools for YARA is transforming the experience of analysts. Innovations like syntax-highlighting editors and the ability to generate rules through intuitive user interfaces are lowering the barrier to entry for creating sophisticated YARA rules. These tools, alongside webinars and educational resources provided by field experts, are enhancing the community's ability to engage with YARA technology. As the project integrates these innovative tools and continues to evolve, it's poised to offer even greater contributions to the cybersecurity community, reflecting a dynamic and responsive approach to the ongoing battle against malware.

CHAPTER-10
CONCLUSION

In concluding the YarWeb project report, it's evident that this initiative marks a transformative advancement in cybersecurity. The project's integration of cutting-edge front-end and back-end technologies, including HTML, CSS, JavaScript, Flask, and Python, has created a powerful tool for the efficient generation and validation of YARA rules. The user-friendly interface and interactive experience it offers for generating rules and analyzing files demonstrate a keen focus on user engagement and functionality.

At its core, YarWeb is a testament to innovative solutions in combating cyber threats. The project's commitment to continuous development, adhering to the best practices of software engineering and incorporating valuable user feedback, ensures its adaptability and relevance in the rapidly evolving cybersecurity landscape. Regular updates to the GUI, website, and the inclusion of new malware signature databases highlight the project's dedication to staying at the forefront of cybersecurity challenges.

YarWeb's scalable and extensible architecture allows for future enhancements and integration with other cybersecurity tools, further enriching its capabilities. This project is not just a standalone tool but a significant contributor to the cybersecurity community. It fosters collaboration and intelligence sharing, positioning itself as a critical asset in the domain of cyber defense.

In conclusion, the YarWeb project stands as a significant milestone in cybersecurity, innovatively addressing the challenges in malware detection and analysis. Its integration of advanced front-end and back-end technologies,

coupled with a focus on user experience and functionality, makes it a powerful tool in the fight against cyber threats. The project's commitment to continuous improvement, guided by best practices and user feedback, ensures its ongoing relevance and effectiveness in the ever-evolving landscape of cybersecurity. YarWeb is more than just a tool; it's a dynamic solution and a testament to the potential of technology in enhancing cyber defense mechanisms.

REFERENCES

1. David Bernal Michelena. “Detecting Malicious Files with YARA Rules as They Traverse the Network using”.

2. Michael Brenge, Christian Rossow. “YARIX: Scalable YARA-based Malware Intelligence” CISPA Helmholtz Centre for Information Security.

3. Nitin Naik, Paul Jenkins, Roger Cooke, Jonathan Gillett, and Yaochu Jin. "Evaluating Automatically Generated YARA Rules and Enhancing Their Effectiveness".

4. Dominika Regeciova, Dusan Kolar, and Marek Milkovic. "Pattern Matching in YARA: Improved Aho-Corasick Algorithm".

5. Nitin Naik, Paul Jenkins, Nick Savage, Longzhi Yang3, Kshirasagar Naik4 and Jingping Song "Augmented YARA Rules Fused with Fuzzy Hashing in Ransomware Triaging”.

6. “Generation of Static YARA-Signatures Using Genetic Algorithm" by Dominika Regeciova, Dusan Kolar, and Marek Milkovic.

7. Automatic YARA Rule Generation" by Myra Khalid, Maliha Ismail, Mureed Hussain, and Muhammad Hanif Durad.

8. Fuzzy Hashing Aided Enhanced YARA Rules" by Nitin Naik, Paul Jenkins, Nick Savage, Longzhi Yang, Kshirasagar Naik, Jingping Song, Tossapon Boongoen, and Natthakan Iam-On

9. Khalid, M., Ismail, M., Hussain, M., and Hanif Durad, M. (2020). "Automatic YARA Rule Generation", 2020 International Conference on Cyber Warfare and Security (ICCWS). doi:10.1109/iccws48432.2020.92923

10. E. Raff, R. Zak, G. L. Munoz, W. Fleming, H. S. Anderson, B. Filar, C. Nicholas, and J. Holt, "Automatic Yara Rule Generation Using Biclustering", https://doi.org/10.48550/arXiv.2009.03779

11. N. Naik, P. Jenkins, R. Cooke, J. Gillett, and Y. Jin, "Evaluating automatically generated YARA rules and enhancing their effectiveness," in Proc. IEEE Symp. Ser. Comput. Intell. (SSCI), Dec. 2020, pp. 1146–1153.

12. N. Naik, P. Jenkins, N. Savage, L. Yang, K. Naik, and J. Song, "Embedding fuzzy rules with YARA rules for performance optimization of malware analysis," in IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). IEEE, 2020.

13. VirusTotal. (2019) YARA in a nutshell. [Online]. Available: https://virustotal.github.io/yara/

14. V. Alvarez. (2019) Writing YARA rules. [Online]. Available: https://yara.readthedocs.io/en/v3.4.0/writingrules.html

15. Readthedocs. (2019) Writing YARA rules. [Online]. Available: https://yara.readthedocs.io/en/v3.5.0/writingrules.html

APPENDIX-A

PSUEDOCODE

A.1 Python Code

```
from flask import Flask, render_template, request, jsonify, send_from_directory, session
import pandas as pd
from flask_sqlalchemy import SQLAlchemy
import bcrypt
import os
import yara
import secrets
import requests
import time
from datetime import datetime
from flask import redirect, url_for
from flask import make_response

app = Flask(__name__)
app.secret_key = secrets.token_hex(16)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///users.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)
with app.app_context():
    db.drop_all()  # Drops all tables
    db.create_all()  # Creates all tables

# Define a User model
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password_hash = db.Column(db.String(128), nullable=False)
    logins = db.Column(db.Integer, default=0)
    last_login_at = db.Column(db.DateTime, default=None)
    total_time_spent = db.Column(db.Integer, default=0)  # Time spent in seconds

    def __repr__(self):
        return '<User %r>' % self.email

# Create the database tables
with app.app_context():
    db.create_all()

yara_directory = "yara_files"
os.makedirs(yara_directory, exist_ok=True)

@app.route('/')
def index():
    return render_template('demo.html')

@app.route('/demo.html')
def index_repeat():
    return render_template('demo.html')

@app.route('/login.html')
def login():
    if 'logged_in' in session and session['logged_in']:
        return redirect(url_for('page2'))
    response = make_response(render_template('login.html'))
    response.headers["Cache-Control"] = "no-cache, no-store, must-revalidate"
    response.headers["Pragma"] = "no-cache"
    response.headers["Expires"] = "0"
    return response

@app.route('/logout')
def logout():
    if 'user_id' in session:
        user = User.query.get(session['user_id'])
        if user and user.last_login_at:
            time_spent = (datetime.now() - user.last_login_at).seconds
            user.total_time_spent += time_spent
            db.session.commit()
    session.pop('logged_in', None)  # Clear the session
    return redirect(url_for('login'))

@app.route('/page2.html')
def page2():
    if 'logged_in' not in session or not session.get('logged_in'):
        return redirect(url_for('login'))
    response = make_response(render_template('page2.html'))
    response.headers["Cache-Control"] = "no-cache, no-store, must-revalidate"
    response.headers["Pragma"] = "no-cache"
    response.headers["Expires"] = "0"
    return response

@app.route('/index.html')
def index1():
    return render_template('index.html')
```

```
@app.route('/test.html')
def test():
    return render_template('test.html')

@app.route('/register', methods=['POST'])
def register():
    email = request.form.get('reg-email')
    password = request.form.get('reg-password').encode('utf-8')
    confirm_password = request.form.get('reg-confirm-password')

    if password.decode('utf-8') != confirm_password:
        return "Passwords do not match. Please try again.", 400

    existing_user = User.query.filter_by(email=email).first()
    if existing_user:
        return "User already exists. Please login.", 409

    # Hashing the password
    hashed_password = bcrypt.hashpw(password, bcrypt.gensalt())

    new_user = User(email=email, password_hash=hashed_password)
    db.session.add(new_user)
    db.session.commit()

    return redirect(url_for('login'))

@app.route('/login', methods=['POST'])
def login_user():
    email = request.form.get('email')
    password = request.form.get('password')

    # Admin credentials (store these securely, for example in environment variables)
    admin_email = "admin@email.com"
    admin_password = "admin1234"

    # Check if user is admin
    if email == admin_email and password == admin_password:
        # If admin login is successful, set the admin session and redirect to dashboard
        session['logged_in'] = True
        session['is_admin'] = True
        return redirect(url_for('dashboard'))

    # Check for regular user login
    user = User.query.filter_by(email=email).first()
    if user and bcrypt.checkpw(password.encode('utf-8'), user.password_hash):
        user.logins += 1
        user.last_login_at = datetime.now()
        db.session.commit()
        session['logged_in'] = True  # Set session variable
        session['user_id'] = user.id  # Store the user's id in the session
        return redirect(url_for('page2'))
    else:
        # Login failed
        return "Invalid credentials. Please try again.", 401


@app.route('/dashboard')
def dashboard():
    if 'logged_in' in session and session['is_admin']:
        users = User.query.all()
        user_data = [{
            'username': user.email,
            'logins': user.logins,
            'time_spent': user.total_time_spent
        } for user in users]
        return render_template('dashboard.html', users=user_data)
    else:
        return redirect(url_for('login'))

@app.route('/update_yara_rule', methods=['POST'])
def update_yara_rule():
    if 'logged_in' not in session or not session.get('is_admin'):
        return redirect(url_for('login'))

    signature = request.form.get('signature')
    string_file = request.files.get('string_file')

    if string_file and signature:
        new_content = string_file.read().decode('utf-8')
        yara_filename = os.path.join(yara_directory, f"{signature}.yara")

        with open(yara_filename, 'r+') as file:
            file_content = file.readlines()
            condition_line_index = next(i for i, line in enumerate(file_content) if line.strip().startswith('condition:'))

            # Calculate the indentation based on the existing condition line
            condition_indent = file_content[condition_line_index].find('condition:')
            indents = ' ' * condition_indent

            # Insert the new content with proper alignment (8 spaces)
            indented_new_content = ' ' * 8 + new_content.replace('\n', f'\n{indents}')
            file_content.insert(condition_line_index, indented_new_content + "\n")
```

```
            file.seek(0)  # Go back to the start of the file
            file.writelines(file_content)  # Write the modified content

        return redirect(url_for('dashboard'))  # Redirect back to the dashboard after updating

    return 'Error updating YARA rule', 400

@app.route('/page2')
def page2_view():
    return render_template('page2.html')

@app.route('/generate', methods=['POST'])
def generate_yara():
    malware_type = request.form.get('malwareType')
    csv_files = {
        "AgentTesla": "AgentTeslaDb.csv",
        "SnakeKeylogger": "SnakeKeyloggerDb.csv",
        "RedLineStealer": "RedLineStealerDb.csv",
        "Loki": "LokiDb.csv"
    }

    file_path = csv_files.get(malware_type, "default.csv")
    yara_rule = read_and_process_csv(file_path, malware_type)

    yara_filename = f"{malware_type}.yara"
    with open(os.path.join(yara_directory, yara_filename), 'w') as file:
        file.write(yara_rule)

    session['yara_filename'] = yara_filename
    return jsonify({'filename': yara_filename})

@app.route('/download/<filename>')
def download_file(filename):
    return send_from_directory(yara_directory, filename, as_attachment=True)

def read_and_process_csv(file_path, malware_type):
    csv_data = pd.read_csv(file_path)

    def deduplicate_and_process_strings(strings):
        unique_strings = set()
        processed_strings = []
        counter = 0

        for string in strings:
            parts = string.split('$s')[1:]
            for part in parts:
                string_start = part.find(' "')
                if string_start != -1:
                    actual_string = part[string_start + 1:].strip()
                    if actual_string not in unique_strings:
                        unique_strings.add(actual_string)
                        processed_strings.append(f"$s{counter} = {actual_string}")
                        counter += 1

        return processed_strings

    deduplicated_processed_strings = deduplicate_and_process_strings(csv_data['Malicious_strings'])

    yara_rule = "rule MalwareDetection {\n"
    yara_rule += "    meta:\n"
    yara_rule += f"        description = \"Generic rule for {malware_type} .exe malwares\"\n"
    yara_rule += "        author = \"Group project\"\n"
    yara_rule += "    strings:\n"
    for s in deduplicated_processed_strings:
        yara_rule += f"        {s}\n"
    yara_rule += "    condition:\n"
    yara_rule += "        ( uint16(0) == 0x5a4d and filesize < 3000KB and ( 8 of them )) or ( all of them )\n"
    yara_rule += "}"

    return yara_rule

@app.route('/test_yara', methods=['POST'])
def test_yara():
    if 'exeFile' not in request.files:
        return 'No file uploaded', 400

    file = request.files['exeFile']
    if file.filename == '':
        return 'No file selected', 400

    filepath = os.path.join('temp', file.filename)
    file.save(filepath)

    # YARA rule matching
    if 'yara_filename' not in session:
        os.remove(filepath)
        return 'YARA rule not generated', 400
```

```
    yara_rule_path = os.path.join(yara_directory, session['yara_filename'])
    try:
        rules = yara.compile(filepath=yara_rule_path)
        matches = rules.match(filepath)
        yara_result = 'Matched rules: ' + ', '.join([match.rule for match in matches]) if matches else 'No matches found'
    except Exception as e:
        os.remove(filepath)
        return f'Error testing YARA rule: {e}', 500

    # VirusTotal API integration
    vt_url = "https://www.virustotal.com/api/v3/files"
    api_key = "41f3945d95b4b36bfae1e72dac3abd998422e306f191d9e429208b69ec0f44ff"

    with open(filepath, 'rb') as f:
        files = {'file': (file.filename, f)}
        headers = {'x-apikey': api_key}
        response = requests.post(vt_url, files=files, headers=headers)

    if response.status_code == 200:
        data = response.json()
        analysis_id = data['data']['id']
        analysis_url = f"https://www.virustotal.com/api/v3/analyses/{analysis_id}"

        # Wait for a few seconds before requesting the results
        time.sleep(15)

        analysis_response = requests.get(analysis_url, headers=headers)
        if analysis_response.status_code == 200:
            analysis_data = analysis_response.json()
            os.remove(filepath)  # Clean up the uploaded file after getting the results
            return jsonify({'yara_result': yara_result, 'vt_data': analysis_data})
        else:
            os.remove(filepath)  # Clean up the uploaded file in case of error
            return jsonify({'yara_result': yara_result, 'vt_data': 'Error retrieving analysis results'})
    else:
        os.remove(filepath)  # Clean up the uploaded file in case of error
        return jsonify({'yara_result': yara_result, 'vt_data': 'Error submitting file to VirusTotal'})




if __name__ == '__main__':
    app.config['DEBUG'] = True
    app.run(host='localhost', port=5000, debug=True)
```

A.2 Yara Rule

```
rule MalwareDetection {
    meta:
        description = "Generic rule for Loki .exe malwares"
        author = "Group project"
    strings:
        $s0 = "lSystem.Resources.ResourceReader, mscorlib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089#System.Resources.R" ascii
        $s1 = "UXlsZakz.exe" fullword wide
        $s2 = "ExecuteButton" fullword wide
        $s3 = "ExecuteButton_Click" fullword ascii
        $s4 = "ProcessOneLine" fullword ascii
        $s5 = "UXlsZakz.pdb" fullword ascii
        $s6 = "PasswordTextbox" fullword wide
        $s7 = "StringCommandParameter" fullword ascii
        $s8 = "BASIC AUTH requires a password" fullword wide
        $s9 = "Executing ..." fullword wide
        $s10 = "HttpMethodComboBox" fullword wide
        $s11 = "get_tSCCgXXP" fullword ascii
        $s12 = "afterAt" fullword ascii
        $s13 = "7453434367585850%716562" fullword wide /* hex encoded string 'tSCCgXXPqeb' */
        $s14 = "MIME (Content Type)" fullword wide
        $s15 = "paramter" fullword ascii
        $s16 = "UsernameLabel" fullword wide
        $s17 = "MarkdownTableFormatter.Properties.Resources.resources" fullword ascii
        $s18 = "ServiceURLTextbox" fullword wide
        $s19 = "ServiceURLLabel" fullword wide
        $s20 = "hSystem.Drawing.Bitmap, System.Drawing, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3aPADPADP" fullword ascii
        $s21 = "HscA.exe" fullword wide
        $s22 = "HscA.pdb" fullword ascii
        $s23 = "GetFreeSquares" fullword ascii
        $s24 = "get_NoSolutions" fullword ascii
        $s25 = "get_Searching" fullword ascii
        $s26 = "* W|-{H" fullword ascii
        $s27 = "get_FoundSolutions" fullword ascii
        $s28 = "75444246%626F69" fullword wide /* hex encoded string 'uDBFboi' */
        $s29 = "*jRjCWRUNx" fullword ascii
        $s30 = "nQueensInput_KeyPress" fullword ascii
        $s31 = "16.0.0.0" fullword ascii
        $s32 = "Win Forms Collaboration" fullword wide
        $s33 = "get_uDBF" fullword ascii
        $s34 = "Eighter" fullword wide
        $s35 = "etna forma aplikacije" fullword wide
        $s36 = "label01" fullword wide
        $s37 = "get_QueenLocations" fullword ascii
        $s38 = "btnStudent7" fullword wide
        $s39 = "SELECT encryptedUsername, encryptedPassword, formSubmitURL, hostname FROM moz_logins" fullword ascii
        $s40 = "sCrypt32.dll" fullword wide
        $s41 = "SmtpPassword" fullword wide
        $s42 = "SMTP Password" fullword wide
        $s43 = "FtpPassword" fullword wide
        $s44 = "%s\\%s%i\\data\\settings\\ftpProfiles-j.jsd" fullword wide
        $s45 = "aPLib v1.01  -  the smaller the better :)" fullword ascii
        $s46 = "%s\\%s\\User Data\\Default\\Login Data" fullword wide
        $s47 = "%s%s\\Login Data" fullword wide
        $s48 = "%s%s\\Default\\Login Data" fullword wide
```

```
        $s49 = "%s\\32BitFtp.TMP" fullword wide
        $s50 = "%s\\GoFTP\\settings\\Connections.txt" fullword wide
        $s51 = "Software\\Microsoft\\Windows NT\\CurrentVersion\\Windows Messaging Subsystem\\Profiles\\Outlook" fullword wide
        $s52 = "%s\\Mozilla\\SeaMonkey\\Profiles\\%s" fullword wide
        $s53 = "%s\\%s\\%s.exe" fullword wide
        $s54 = "More information: http://www.ibsensoftware.com/" fullword ascii
        $s55 = "%s\\nss3.dll" fullword wide
        $s56 = "PopPassword" fullword wide
        $s57 = "SmtpPort" fullword wide
        $s58 = "SmtpAccount" fullword wide
        $s59 = "MKiJ887777.exe" fullword wide
        $s60 = "       <requestedExecutionLevel level=\"asInvoker\" uiAccess=\"false\"/>" fullword ascii
        $s61 = "  <assemblyIdentity version=\"1.0.0.0\" name=\"MyApplication.app\"/>" fullword ascii
        $s62 = " Type Descriptor'" fullword ascii
        $s63 = " constructor or from DllMain." fullword ascii
        $s64 = "  <trustInfo xmlns=\"urn:schemas-microsoft-com:asm.v2\">" fullword ascii
        $s65 = "XQQ.exS" fullword ascii
        $s66 = " Class Hierarchy Descriptor'" fullword ascii
        $s67 = " Base Class Descriptor at (" fullword ascii
        $s68 = " Complete Object Locator'" fullword ascii
        $s69 = "1CXm- " fullword ascii
        $s70 = "     <requestedPrivileges xmlns=\"urn:schemas-microsoft-com:asm.v3\">" fullword ascii
        $s71 = "MKiJ887777" fullword wide
        $s72 = "Broken pipe" fullword ascii /* Goodware String - occured 742 times */
        $s73 = "Permission denied" fullword ascii /* Goodware String - occured 823 times */
        $s74 = "c;XxiVu^-" fullword ascii
        $s75 = "SQKL4wp" fullword ascii
        $s76 = "L$PQSV" fullword ascii /* Goodware String - occured 1 times */
        $s77 = "CZBj>T]e];.n" fullword ascii
        $s78 = "9pMKrVn4" fullword ascii
    condition:
        ( uint16(0) == 0x5a4d and filesize < 3000KB and ( 8 of them )) or ( all of them )
}
```

APPENDIX-B

SCREENSHOTS

B.1 Website

Fig B.1.1 Home page

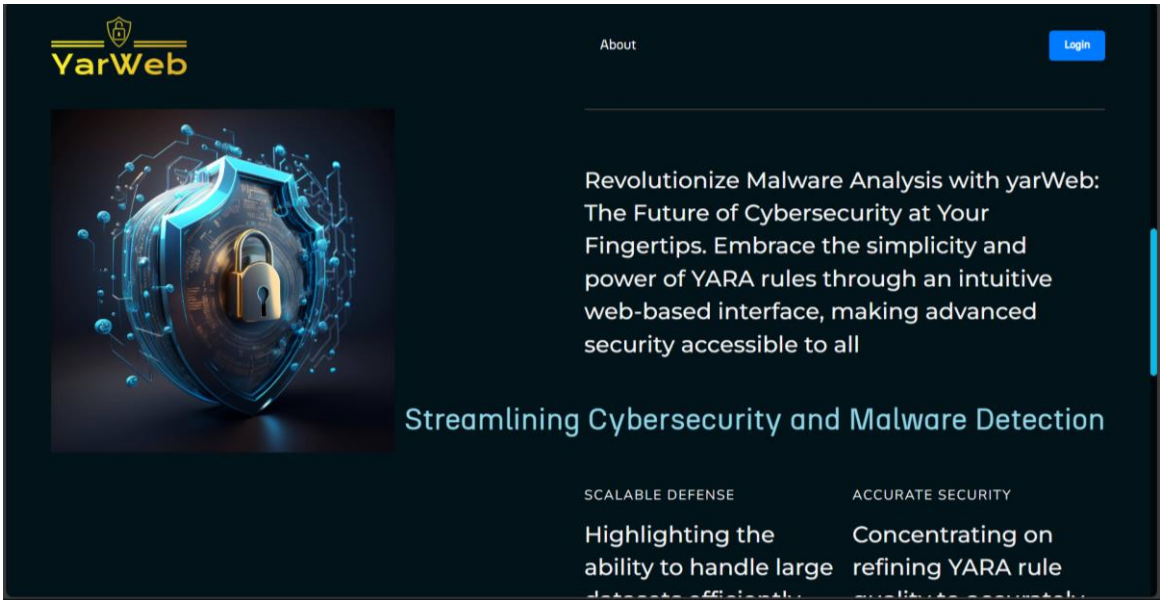

Fig B.1.2 Home page

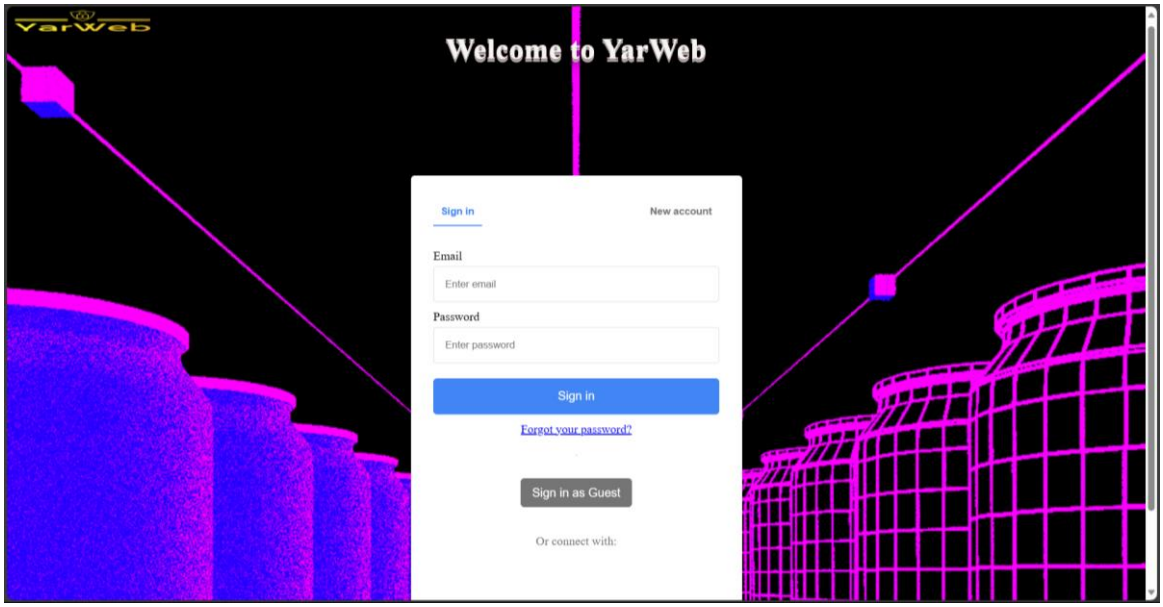

Fig B.1.3 Login page

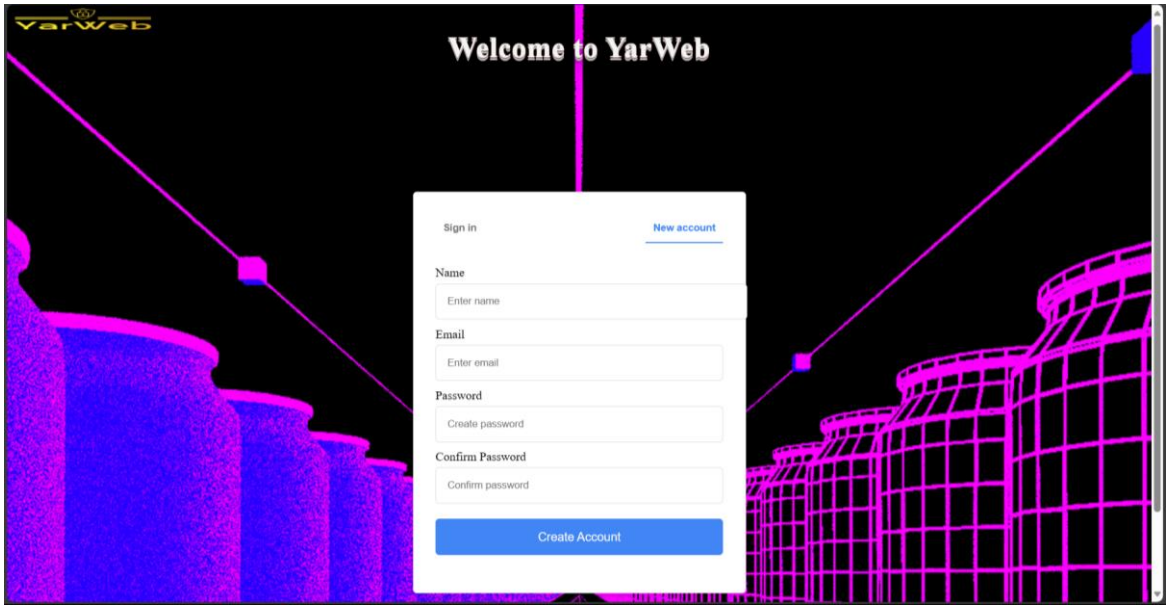

Fig B.1.4 Registration page

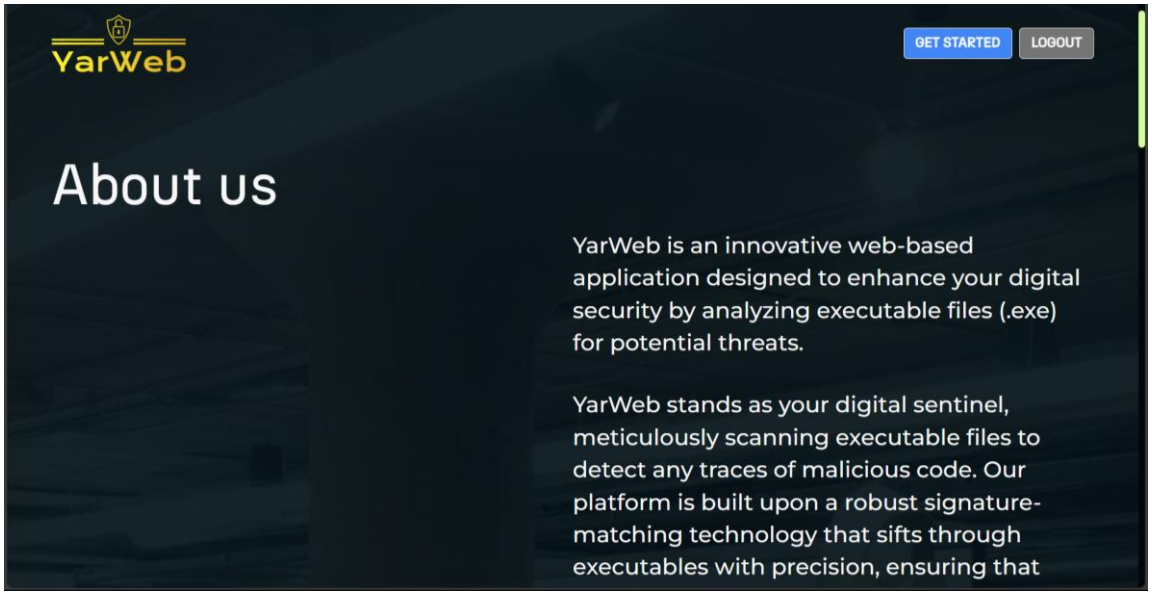

Fig B.1.5 About us page

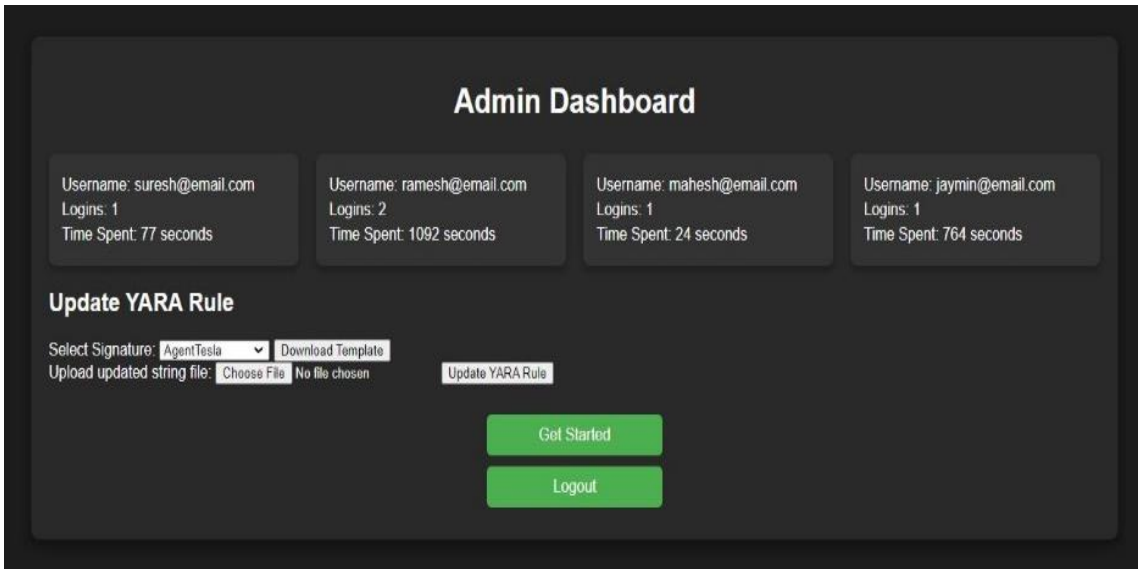

Fig B.1.6 Admin dashboard

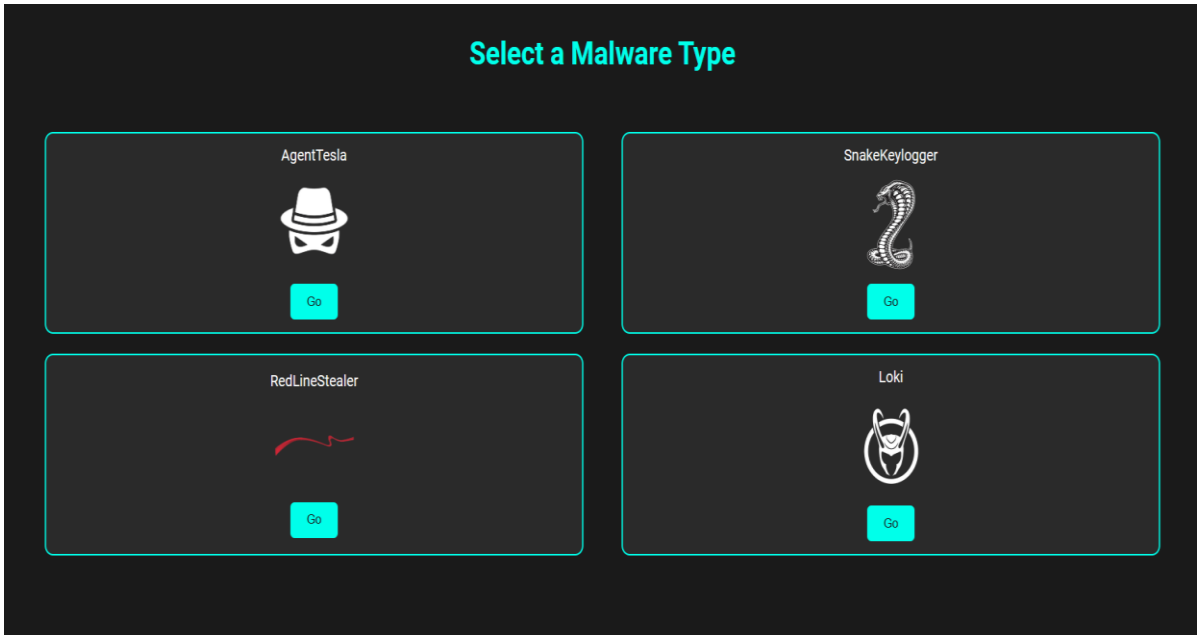

Fig B.1.7 Types of malwares

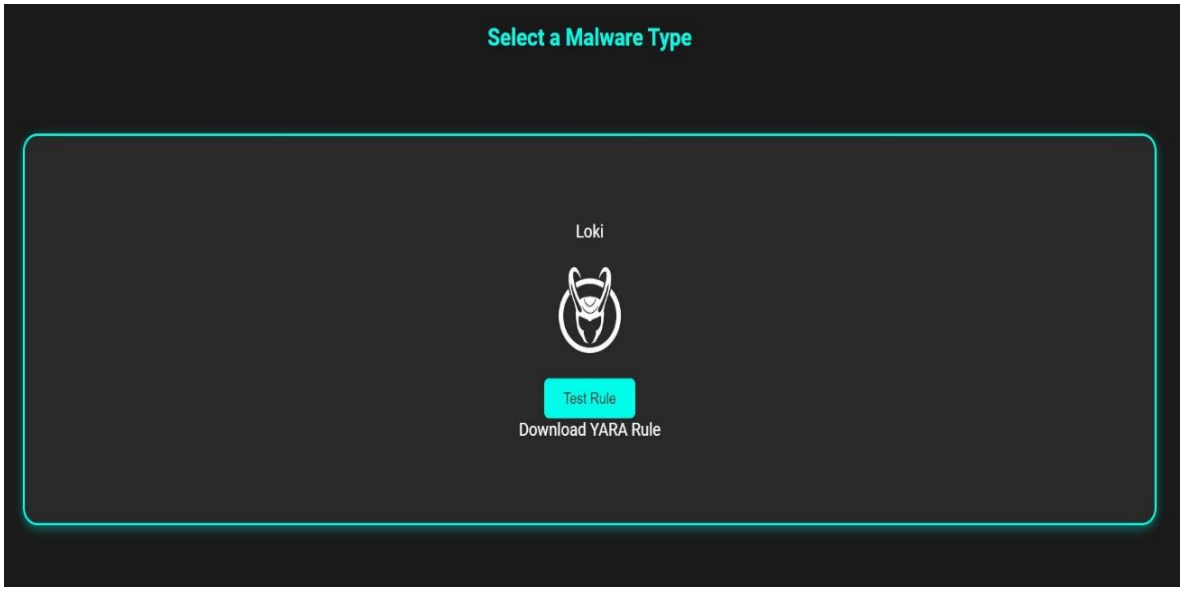

Fig B.1.8 Download Yara rule

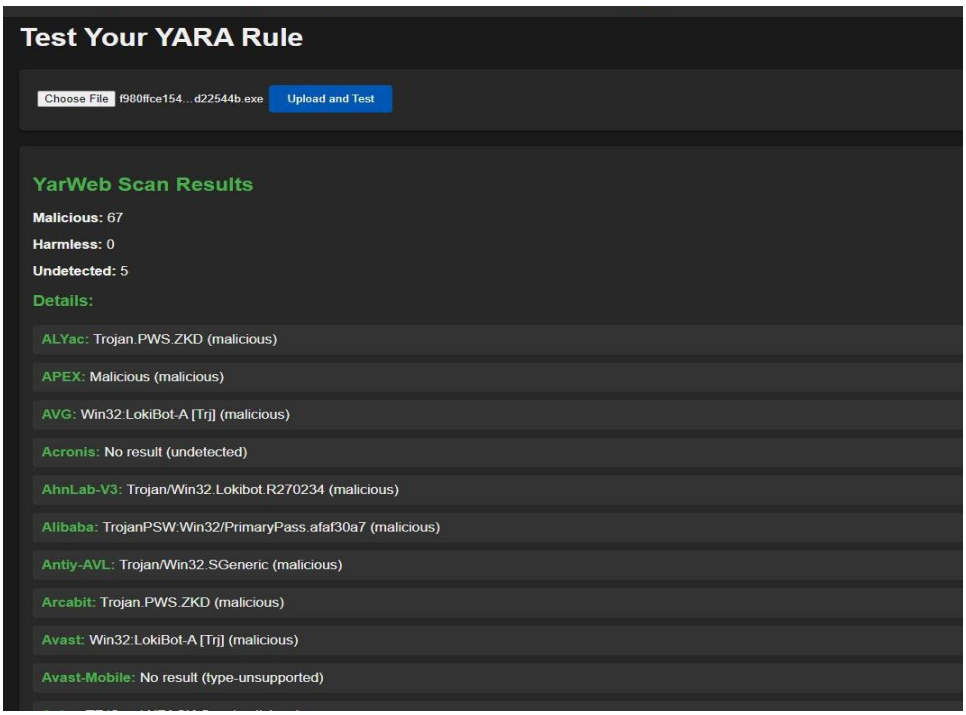

Fig B.1.9 Upload a file

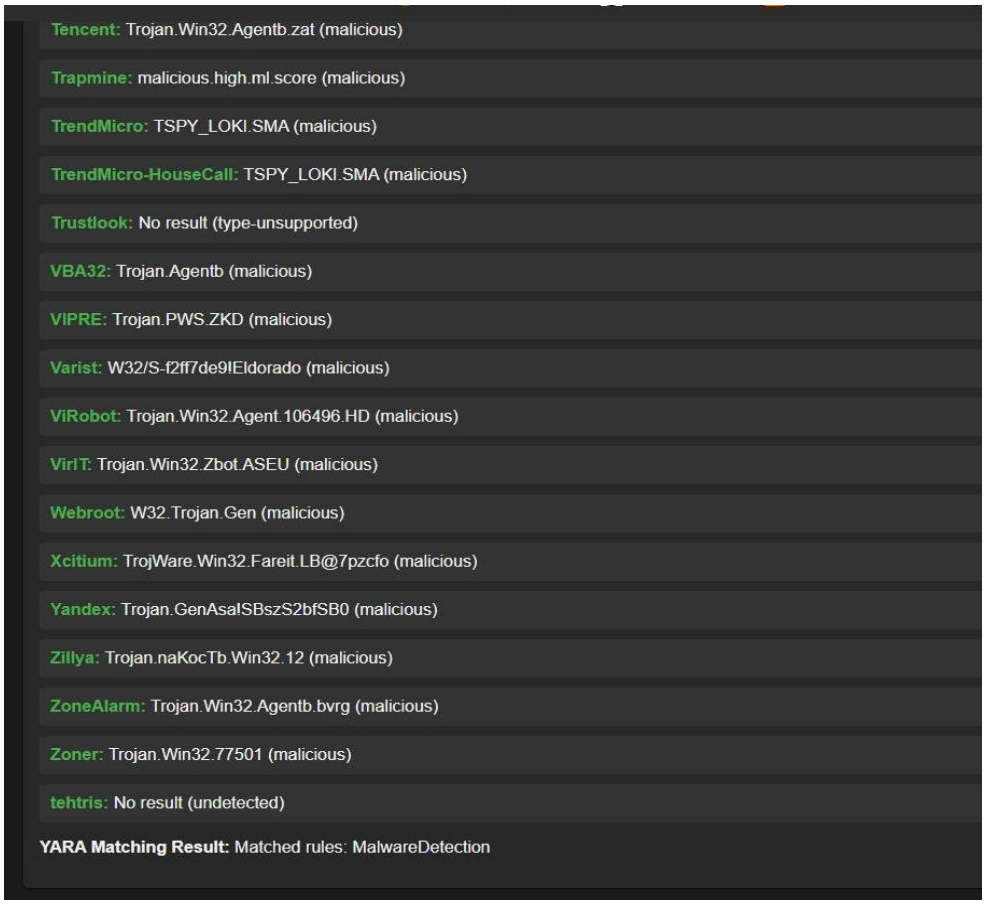

Fig B.1.10 Scanned Results

APPENDIX-C

ENCLOSURES

1. Similarity Index / Plagiarism Check report clearly showing the Percentage (%).

ORIGINALITY REPORT

22%	20%	16%	16%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Presidency University Student Paper	9%
2	Submitted to M S Ramaiah University of Applied Sciences Student Paper	2%
3	publications.aston.ac.uk Internet Source	2%
4	B Varshini, HR Yogesh, Syed Danish Pasha, Maaz Suhail, V Madhumitha, Archana Sasi. "IoT-Enabled Smart Doors for Monitoring Body Temperature and Face Mask Detection", Global Transitions Proceedings, 2021 Publication	1%
5	canada.explore.openaire.eu Internet Source	1%
6	export.arxiv.org Internet Source	1%
7	tigerprints.clemson.edu Internet Source	1%

8	Dominika Regeciova, Dusan Kolar, Marek Milkovic. "Pattern Matching in YARA: Improved Aho-Corasick Algorithm", IEEE Access, 2021 Publication	<1 %
9	journal.cecyf.fr Internet Source	<1 %
10	edevice.fujitsu.com Internet Source	<1 %
11	webthesis.biblio.polito.it Internet Source	<1 %
12	www.nccs.pk Internet Source	<1 %
13	i.blackhat.com Internet Source	<1 %
14	Pragya Bharti, Shreya Saha Roy, A. Suresh. "Implementation of Yara Rules in Android", 2023 International Conference on Computer Communication and Informatics (ICCCI), 2023 Publication	<1 %
15	Submitted to Higher Education Commission Pakistan Student Paper	<1 %
16	www.adaptavist.com Internet Source	<1 %

17	Marcus Botacin, Francis B. Moreira, Philippe O. A. Navaux, André Grégio, Marco A. Z. Alves. " : A Secure Coprocessor to Accelerate Real-Time AntiViruses Using Inspection Breakpoints ", ACM Transactions on Privacy and Security, 2022 Publication	<1 %
18	Submitted to South Bank University Student Paper	<1 %
19	qspace.library.queensu.ca Internet Source	<1 %
20	www.safaribooksonline.com Internet Source	<1 %
21	zerply.com Internet Source	<1 %
22	acikbilim.yok.gov.tr Internet Source	<1 %
23	researchportal.northumbria.ac.uk Internet Source	<1 %
24	swcarpentry.github.io Internet Source	<1 %
25	www.politesi.polimi.it Internet Source	<1 %
26	www.researchgate.net Internet Source	<1 %

27	www.slideshare.net Internet Source	<1 %
28	Hong Jiang, Qingsong Yu, Zhiyue Zhang. "An Improved Quad-Array Trie Algorithm for Website Sensitive Word Detection", Proceedings of the 2023 9th International Conference on Computing and Artificial Intelligence, 2023 Publication	<1 %
29	Hsiang-Yu Chuang, Jiann-Liang Chen, Yi-Wei Ma. "Malware Detection and Classification Based on Graph Convolutional Networks and Function Call Graphs", IT Professional, 2023 Publication	<1 %
30	Nitin Naik, Paul Jenkins, Nick Savage, Longzhi Yang, Tossapon Boongoen, Natthakan Iam-On, Kshirasagar Naik, Jingping Song. "Embedded YARA rules: strengthening YARA rules utilising fuzzy hashing and fuzzy rules for malware analysis", Complex & Intelligent Systems, 2020 Publication	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		

2. Conference Paper

431	**YarWeb: "Web-based Generic YARA Rule Generator"** Show abstract	Track 2 VLSI & Embedded Systems, IoT and Computational Intelligence Systems Email Track Chair	**Submission files:** YarWeb_IEEE.pdf	**Submission:** Edit Submission Edit Conflicts Delete Submission **Supplementary Material:** Upload Supplementary Material

Waiting for acceptance

3. The Project work carried out here is mapped to SDG-9: Industry, Innovation, and Infrastructure.

The YarWeb project embodies the spirit of Sustainable Development Goal 9, focusing on building resilient infrastructure. By advancing cybersecurity technologies, it enhances digital infrastructure's robustness, crucial in our technology-driven era.

This project underlines the importance of sustainable industrialization in the digital domain. Its innovative approach in cybersecurity contributes to creating safer, more reliable digital systems, which are essential for modern industry and commerce.

YarWeb's contribution to fostering innovation is significant. It represents a leap in securing digital networks and data, playing a vital role in the global movement towards secure, resilient, and sustainable technological progress.