

A Project report on

Code Guardian Pro

A Dissertation submitted to JNTU Hyderabad in partial fulfillment of the academic requirements for the award of the degree.

Bachelor of Technology
in
Computer Science and Engineering

Submitted by

GOPU CHITRA BHANU REDDY
(20H51A05C4)

JAKKIDI SANTHOSH REDDY
(20H51A05E1)

TWINKLE SHARMA
(20H51A05Q3)

Under the esteemed guidance of

Mr. V. NARASIMHA
(Assistant Professor)



Department of Computer Science and Engineering

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(UGC Autonomous)

*Approved by AICTE *Affiliated to JNTUH *NAAC Accredited with A⁺ Grade

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD - 501401.

2020- 2024

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

This is to certify that the Major Project entitled "**Code Guardian Pro**" being submitted by Gopu Chitra Bhanu Reddy (20H51A05C4), Jakkidi Santhosh Reddy (20H51A05E1), Twinkle Sharma (20H51A05Q3) in partial fulfillment for the award of **Bachelor of Technology in Computer Science and Engineering** is a record of bonafide work carried out under my guidance and supervision.

The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree.

Mr. V. Narasimha
Assistant Professor

Dr. Siva Skandha Sanagala
Associate Professor and HOD
Dept. of CSE

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

With great pleasure we want to take this opportunity to express my heartfelt gratitude to all the people who helped in making this project work a grand success.

We are grateful to our guide **Mr. V. Narasimha**, Assistant Professor, Department of Computer Science and Engineering & Principal for his valuable technical suggestions and guidance during the execution of this project work.

We would like to thank **Dr. Siva Skandha Sanagala**, Head of the Department of Computer Science and Engineering, CMR College of Engineering and Technology, who is the major driving force to complete our project work successfully.

We are very grateful to **Dr. Ghanta Devadasu**, Dean-Academics, CMR College of Engineering and Technology, for his constant support and motivation in carrying out the project work successfully.

We are highly indebted to **Major Dr. V.A. Narayana**, Principal, CMR College of Engineering and Technology, for giving permission to carry out this project in a successful and fruitful way.

We would like to thank the **Teaching & Non- teaching** staff of Department of Computer Science and Engineering for their co-operation

We express our sincere thanks to **Shri. Ch. Gopal Reddy**, Secretary, CMR Group of Institutions, and **Shri. Ch. Abhinav Reddy**, CEO, CMR Group of Institutions for their continuous care and support.

Finally, We extend thanks to our parents who stood behind us at different stages of this Project. We sincerely acknowledge and thank all those who gave support directly and indirectly in completion of this project work.

Gopu Chitra Bhanu Reddy	20H51A05C4
Jakkidi Santhosh Reddy	20H51A05E1
Twinkle Sharma	20H51A05Q3

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO.
	LIST OF FIGURES	iii
	ABSTRACT	v
1	INTRODUCTION	1
	1.1 Problem Statement	2
	1.2 Research Objective	3
	1.3 Project Scope and Limitations	4
2	BACKGROUND WORK	5
	2.1 Nmap Network Reconnaissance	6
	2.1.1 Introduction	6
	2.1.2 Merits, Demerits and Challenges	6
	2.1.3 Implementation	8
	2.2 A Comprehensive Overview of Nessus Scanning and Analysis	9
	2.2.1 Introduction	9
	2.2.2 Merits, Demerits and Challenges	10
	2.2.3 Implementation	11
	2.3 Traditional and Automated Phishing Detection Techniques	12
	2.3.1 Introduction	12
	2.3.2 Merits, Demerits and Challenges	12
	2.3.3 Implementation	13
3	PROPOSED SYSTEM	14
	3.1 Objective of Proposed Method	15
	3.2 Methods Used for Proposed Method	16
	3.3 Designing	19
	3.3.1 Use case Diagram	19
	3.3.2 Sequence Diagram	20
	3.3.3 Directory Diagram	21

3.4	Stepwise Implementation and Code	22
3.4.1	Implementation Steps	22
3.4.2	Code	26
3.5	System Architecture	35
3.6	System Requirements	36
3.6.1	Hardware Requirements	36
3.6.2	Software Requirements	36
4	RESULTS AND DISCUSSION	37
4.1	Results and Discussion	38
5	CONCLUSION	47
5.1	Conclusion and Future Enhancement	48
5.2	Future Scope	48
6	REFERENCES	49
7	APPENDIX	53

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
2.1	Sample Diagram of Nmap Network Reconnaissance	9
2.2	Test Report Performed with Nessus Vulnerability Scanner	11
2.3	Comparison with Traditional phishing URL Detection Methods on Data	13
3.1	Objective Diagram of Proposed Method	15
3.2	Pictorial Representation of Personal Information Discovery	17
3.3	Working of Code Guardian Pro Proposed Method	18
3.4	Use Case Diagram	19
3.5	Data Flow Diagram	20
3.6	Directory Diagram	21
3.7	System Architecture	35
4.1	Testing Out the Github Link for the Phishing link	39
4.2	Exploit Scan	41
4.3	Result of Vulnerability Scanning	41
4.4	Syntax Error	42
4.5	Logical Error	43
4.6	SQL Injection Detection	43
4.7	IP Address Lookup	45
4.8	Finding IP Location on Gmaps	45
4.9	Phone Number Lookup	46
4.10	Google Account Data	46
4.11	Gmail Account Data	46

ABSTRACT

The Code Guardian Pro project is an advanced complex tool find bugs and errors in their code as well as the software built and later improved the overall quality and the level of productivity. The tool employs sophisticated parsing of programming language constructs, static code analysis, and identification of patterns to detect bugs both at a syntax level and logical level. The bug warlock in Code Guardian Pro gives more than standard identify errors but also provides briefly but understandable tips on how to correct the identified errors. In addition to this, the Code Inspection and Security Analysis tools are also able to conduct detailed vulnerability analysis, exposing critical security risks found within the source code. Not only it goes through the source codes, but it can also identify devious patterns in websites, emails and even may detect errors in the images. On the other hand, the Code Guardian Pro naturally expedites the debugging, decreases the development time, and increases the robustness, security, and reliability of software applications regardless of the size of the potential bugs and issues. Consequently, the end product is a better user experience.

CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

In the landscape of software development and cybersecurity, there exists a critical need for advanced tools capable of efficiently detecting and rectifying bugs, errors, and vulnerabilities within codebases and software applications [1][2]. While various vulnerability scanners and intrusion detection systems have been proposed in the literature, there remains a gap in the availability of comprehensive solutions that seamlessly integrate sophisticated parsing techniques, static code analysis, and pattern recognition algorithms to identify and mitigate both syntax-level and logic-level issues [3][4].

Additionally, existing approaches often lack in providing actionable insights for developers to address identified errors effectively [5]. Furthermore, the prevalence of security threats such as SQL injection attacks, phishing attempts, and adversarial machine learning poses significant challenges to the integrity and reliability of software systems, necessitating robust defense mechanisms [6][7]." Code Guardian Pro's capability to conduct detailed vulnerability analysis and its versatility in detecting devious patterns across various digital landscapes, including websites, emails, and even images, positions it as a vital tool in fortifying the security posture of software systems.

Therefore, the problem statement revolves around the development of Code Guardian Pro, an advanced tool aimed at enhancing the quality, productivity, and security of software applications by offering comprehensive bug detection, error identification, and vulnerability analysis capabilities. The project seeks to address the shortcomings of existing solutions by providing developers with actionable insights to rectify identified issues promptly, thereby expediting the debugging process and bolstering the overall robustness and reliability of software systems in the face of evolving cybersecurity threats.

1.2 Research Objective

The primary aim of this research is to develop an advanced software tool that addresses the inherent challenges in bug detection and code quality improvement within the software development lifecycle. Traditional methods often fall short in identifying all possible issues, leading to subpar software quality and heightened security risks. Therefore, the research seeks to create an improved tool that streamlines the process of identifying and rectifying bugs, syntax errors, logical problems, and security vulnerabilities across various domains such as source code, websites, and emails.

This multifaceted software development endeavour encompasses the integration of static code analysis and pattern recognition mechanisms to enhance bug detection and code quality. By leveraging advanced algorithms, the tool aims to uncover both syntax and logic-based issues, thereby ensuring a more comprehensive approach to bug identification. Additionally, the project extends its scope to include thorough threat analysis to expose potential security loopholes within the codebase, spanning programming codes, website content, message formats, and image quality.

However, it is essential to acknowledge the limitations and challenges associated with automated tools like Code Guardian Pro. Despite their efficacy in streamlining processes and enhancing productivity, they may occasionally yield false positives and false negatives, necessitating careful scrutiny from developers. Additionally, compatibility barriers, privacy concerns, and the learning curve associated with such tools need to be addressed to ensure widespread adoption and successful integration into both large and small development teams. Consequently, the research aims to provide insights into mitigating these challenges and optimizing the utilization of automated tools for bug detection and code quality improvement in software development.

1.3 Project Scope and Limitations

Scope:

1. **Comprehensive bug detection:** The tool aims to detect bugs and errors in code, software applications, websites, emails, and even images. It employs sophisticated parsing techniques, static code analysis, and pattern recognition to identify issues at both syntax and logical levels.
2. **Code quality improvement:** It not only identifies errors but also provides actionable tips on how to correct them, thereby enhancing the overall quality of the codebase.
3. **Security analysis:** The tool conducts detailed vulnerability analysis, exposing critical security risks within the source code, websites, and other digital assets.
4. **Enhanced productivity:** By expediting the debugging process and decreasing development time, Code Guardian Pro aims to increase productivity and efficiency in software development.
5. **Improved user experience:** Ultimately, the project seeks to deliver software applications that are more robust, secure, and reliable, leading to a better user experience for end-users.

Limitations

1. **False positives and false negatives:** Like any automated tool, it may occasionally produce inaccurate results, leading to false positives (incorrectly identifying issues) or false negatives (failing to detect actual issues).
2. **Human expertise required:** While Code Guardian Pro provides guidance for bug identification and correction, human expertise is still essential for making complex decisions, particularly in dealing with intricate code or unusual methods.
3. **Compatibility and privacy concerns:** The tool may face compatibility barriers with certain software environments, and concerns about privacy infringement may arise due to the thorough analysis of digital assets.
4. **Learning curve:** Users may require time to familiarize themselves with the tool's functionalities and optimize its usage for their specific needs.

CHAPTER 2

BACKGROUND WORK

CHAPTER 2

BACKGROUND WORK

2.1 Nmap Network Reconnaissance

2.1.1 Introduction

Nmap, short for Network Mapper, is a powerful and widely-used port scanner that provides detailed information about hosts, IP addresses, and network services. It is an essential tool for network administrators, security professionals, and hackers alike, as it allows for comprehensive reconnaissance of network infrastructure. By scanning target systems, Nmap can identify open ports, services running on those ports, and even predict the type of operating system being used[1]. Additionally, it can generate visual representations of network topologies, helping users understand the layout of interconnected machines and gateways.

2.1.2 Merits, Demerits and Challenges

Merits:

- Comprehensive scanning capabilities: Nmap excels at scanning and analyzing network infrastructure, providing detailed information about hosts, IP addresses, ports, and services.
- Predictive OS detection: By analyzing network responses, Nmap can accurately predict the type of operating system running on a target machine, aiding in vulnerability assessment and penetration testing.
- Visual representation of network topology: Nmap generates graphical representations of network topologies, making it easier for users to understand the layout of interconnected machines and gateways.
- Flexible port scanning options: Nmap allows users to scan specific ranges of ports or entire networks, providing flexibility in reconnaissance tasks.
- Open-source and widely supported: Nmap is open-source software, freely available to download and use, and is supported by a large community of users and developers, ensuring regular updates and improvements.

Demerits:

- Legal and ethical concerns: While Nmap is a valuable tool for network reconnaissance, its misuse for unauthorized scanning and hacking activities can have legal and ethical implications.
- False positives: Nmap may occasionally produce false positives, incorrectly identifying open ports or misclassifying operating systems, leading to potential misinformation and wasted resources.
- Resource-intensive: Depending on the scope of the scan and the size of the network, Nmap scans can be resource-intensive and may impact network performance.
- Detection by intrusion detection systems: Nmap scans can be detected and blocked by intrusion detection systems (IDS) and firewall rules, limiting its effectiveness in stealthy reconnaissance operations.
- Limited to network layer scanning: While Nmap provides valuable insights into network infrastructure, it is primarily focused on scanning the network layer and may not detect vulnerabilities or misconfigurations at higher layers, such as application layer vulnerabilities.

Challenges:

- Legal and ethical considerations: Ensuring that the use of Nmap complies with legal regulations and ethical guidelines, as unauthorized scanning can lead to legal consequences and ethical dilemmas.
- Resource constraints: Managing resource-intensive scans, considering the impact on network performance, and efficiently using computing resources to minimize disruption.
- Complexity of network environments: Navigating through complex and dynamic network architectures, devices, and services to conduct effective reconnaissance, requiring adaptability and expertise in network security principles.

2.1.3 Implementation

The implementation of Nmap involves utilizing its powerful port scanning capabilities to gather information about a target IP address or range of IP addresses. When provided with an IP address, Nmap systematically probes the target system to identify open ports, closed ports, and the services associated with those ports. This process involves sending various types of packets to the target and analyzing the responses to determine the status of each port. By examining the responses from the target, Nmap can infer the type of services running on the system, whether they are using TCP, UDP, or other communication protocols. Additionally, Nmap can provide insights into the operating system running on the target machine by analyzing subtle differences in the way the system responds to probing packets.

Furthermore, Nmap can generate a detailed host topology diagram, illustrating the network path to the target machine and highlighting any intermediary gateways or routers. This visual representation helps users understand the network layout and identify potential points of entry for further exploration. Once the network topology is understood, Nmap can be used to conduct more targeted scans, focusing on specific ranges of ports or services. This capability allows attackers to identify potential vulnerabilities and exploit them to gain unauthorized access to the target system, potentially compromising sensitive data or intellectual property. Therefore, while Nmap is a powerful tool for network reconnaissance and analysis, it is essential for organizations to implement robust security measures to protect against unauthorized access and data breaches.

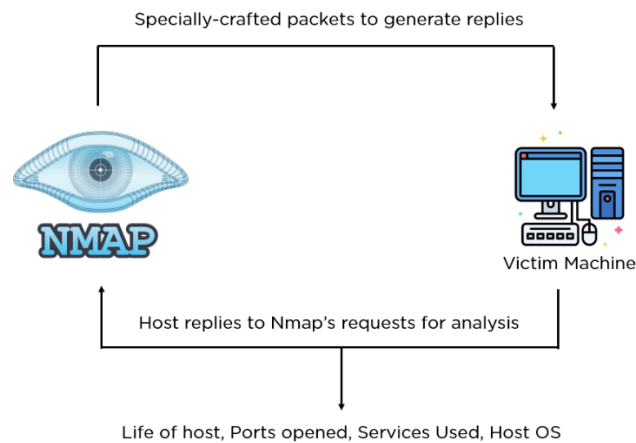


Figure 2.1: Sample diagram of Nmap Network Reconnaissance

2.2 A Comprehensive Overview of Nessus Scanning and Analysis

2.2.1 Introduction

Nessus is a powerful tool for discovering vulnerabilities within remote hosts, offering comprehensive scans that unveil a spectrum of potential security risks [1]. Its deployment facilitates both internal and external scans, providing a thorough examination of networked assets. With an Origin Scan feature leveraging the host computer's router, Nessus effectively identifies hosts within a network, ensuring no potential vulnerability goes unnoticed. Moreover, its application extends to web testing, offering multiple scanning methods including template-based approaches. Nessus boasts the capability to conduct multiple scans simultaneously, enhancing efficiency without compromising accuracy. Through its robust analysis, Nessus categorizes vulnerabilities into four distinct levels, ranging from high severity to informational, enabling users to prioritize remediation efforts effectively [2]. Results are promptly archived upon completion of scans, ensuring accessibility for subsequent analysis and action.

Utilizing a client-server architecture, Nessus optimizes performance while maintaining rigorous validation standards. Leveraging HTTP for monitoring and a dedicated web interface, it seamlessly integrates into existing infrastructures. The client-side initiation of sessions, coupled with server-side validation, ensures reliability and integrity throughout the scanning process. Nessus empowers users to export results in various formats, including CSV and JSON,

facilitating seamless integration into diverse workflows. Furthermore, its comprehensive reports equip stakeholders with actionable insights, streamlining vulnerability management processes. From identifying critical vulnerabilities to ensuring host compliance, Nessus emerges as an indispensable asset in safeguarding digital assets against evolving threats.

2.2.2 Merits, Demerits and Challenges

Merits:

- **Comprehensive Vulnerability Detection:** Nessus is capable of conducting thorough scans to identify vulnerabilities present in both internal and external systems. It covers a wide range of vulnerabilities and provides detailed reports on the issues found, helping organizations prioritize and address security weaknesses effectively.
- **Flexible Reporting Options:** Nessus offers the flexibility to export scan results in various formats such as CSV or JSON. This allows security teams to integrate the findings into their existing workflows and tools easily.
- **Scalability and Parallel Scanning:** Nessus supports parallel scanning, enabling multiple scans to be conducted simultaneously on different hosts or networks. This scalability is beneficial for organizations with large infrastructures, as it helps reduce scanning time and improve overall efficiency in managing security assessments.

Demerits:

- **False positives:** Nessus may sometimes report vulnerabilities that are not actually present, leading to wasted time and resources investigating non-issues.
- **Limited exploitability assessment:** While Nessus identifies vulnerabilities, it may not always accurately assess the likelihood of those vulnerabilities being exploited in a real-world scenario.
- **Resource-intensive:** Nessus scans can consume significant system resources, potentially impacting network performance and causing disruptions to ongoing operations.

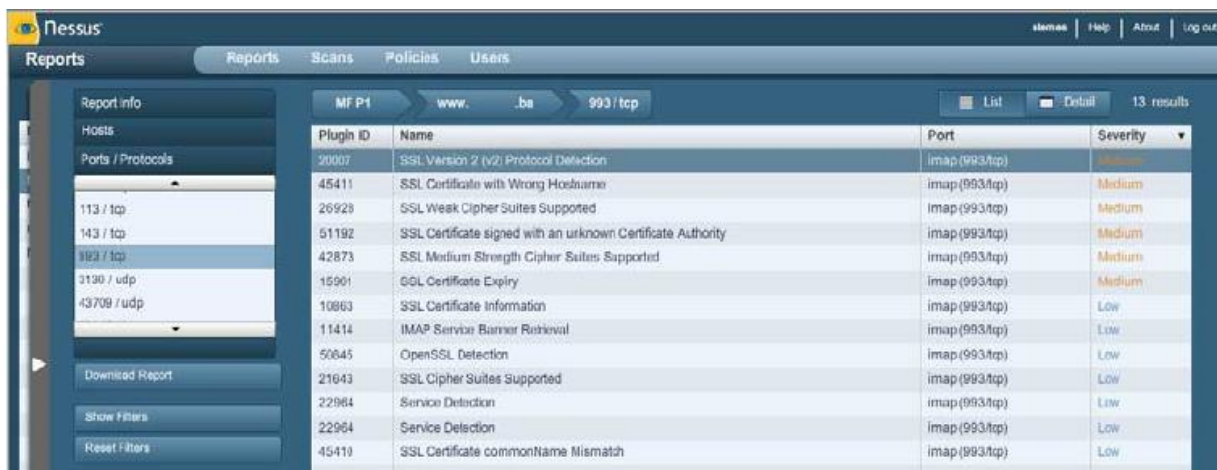
Challenges:

- **Scalability:** Managing large-scale scans and processing results efficiently.
- **False Positives:** Addressing inaccuracies in vulnerability detection to minimize false alarms.

- **Network Complexity:** Navigating intricate network infrastructures to ensure comprehensive scanning coverage.

2.2.3 Implementation

In implementation, Nessus serves as a pivotal tool for proactive cybersecurity measures, offering thorough vulnerability assessment across networks. Its versatility accommodates diverse scanning needs, from internal stem cell scans to web application testing, providing nuanced insights through vulnerability categorization. The seamless integration of client-server architecture streamlines operations, with HTTP facilitating efficient communication between components. Notably, its ability to export results in various formats enhances usability, while stringent validation measures on the server side ensure data integrity. Overall, Nessus empowers organizations to fortify their cybersecurity posture, enabling prompt detection and mitigation of potential threats.



Plugin ID	Name	Port	Severity
20007	SSL Version 2 (v2) Protocol Detection	imap (993/tcp)	Medium
45411	SSL Certificate with Wrong Hostname	imap (993/tcp)	Medium
26928	SSL Weak Cipher Suites Supported	imap (993/tcp)	Medium
51192	SSL Certificate signed with an unknown Certificate Authority	imap (993/tcp)	Medium
42873	SSL Medium Strength Cipher Suites Supported	imap (993/tcp)	Medium
15901	SSL Certificate Expiry	imap (993/tcp)	Medium
10863	SSL Certificate Information	imap (993/tcp)	Low
11414	IMAP Service Banner Retrieval	imap (993/tcp)	Low
50645	OpenSSL Detection	imap (993/tcp)	Low
21643	SSL Cipher Suites Supported	imap (993/tcp)	Low
22964	Service Detection	imap (993/tcp)	Low
22964	Service Detection	imap (993/tcp)	Low
45419	SSL Certificate commonName Mismatch	imap (993/tcp)	Low

Figure 2.2: Test Report Performed with Nessus Vulnerability Scanner

2.3 Comparison of Traditional and Automated Phishing Detection Techniques

2.3.1 Introduction

In their 2018 study, Qabajeh et al. explored the efficacy of traditional and automated approaches in detecting phishing attempts [7]. Traditional methods primarily focus on individual training through workshops and sessions, often emphasizing legal aspects. In contrast, automated anti-phishing techniques encompass list-based and Machine Learning (ML) approaches, offering more systematic and scalable solutions [14]. The comparison between these two methodologies highlights their similarities and delineates positive and negative elements from the user's perspective, ultimately aiming to enhance cybersecurity measures against phishing threats. However, it's worth noting that the research's scope is limited, as it doesn't delve into the realm of Deep Learning methods for online phishing website detection, representing a potential avenue for future exploration and refinement of anti-phishing strategies [5].

2.3.2 Merits, Demerits and Challenges

Merits:

- **Efficiency:** These methods can quickly analyze large volumes of data and adapt to evolving phishing techniques.
- **Accuracy:** Machine learning algorithms can identify subtle patterns and indicators of phishing, enhancing detection accuracy.
- **Scalability:** Automated approaches are easily scalable to protect against a wide range of phishing attacks, reducing the burden on individual users and organizations.

Demerits:

- **Exclusion of Deep Learning methods.**
- **Limited exploration of user-centric perspectives.**
- **Potential bias towards traditional methods over emerging technologies.**

Challenges:

- Incorporating Deep Learning methods for online phishing website detection.
- Addressing the limitations of traditional anti-phishing methods.
- Balancing positive and negative elements of list-based and Machine Learning Based approaches.
- Ensuring user-friendly implementation of automated anti-phishing tools.
- Continuously updating methodologies to adapt to evolving phishing techniques.

2.3.3 Implementation

Traditional methods involve user training through workshops, often emphasizing legal aspects. In contrast, automated anti-phishing methods include list-based and machine learning-based approaches. The study compares these techniques, noting both similarities and user-oriented advantages and disadvantages. Machine learning and rule induction emerge as the most suitable tools for phishing protection, according to the research. However, the study acknowledges limitations, such as its focus on 67 foundational research items and the exclusion of deep learning methods for online phishing detection. Despite these constraints, the research sheds light on effective strategies for combating phishing attacks, underscoring the importance of leveraging advanced technologies like machine learning in cybersecurity efforts.

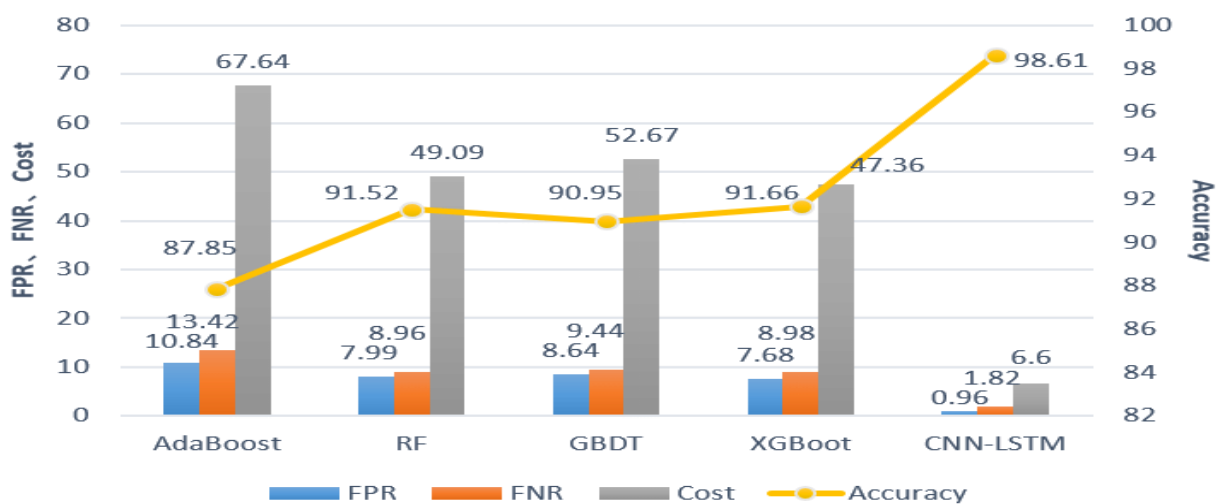


Figure 2.3: Comparison with Traditional Phishing URL Detection Methods on Data

CHAPTER 3

PROPOSED SYSTEM

CHAPTER 3

PROPOSED SYSTEM

3.1 Objectives of Proposed Method

The Code Guardian Pro Project aims to comprehensively evaluate the effectiveness and efficiency of this innovative software quality assurance tool. It focuses on assessing Code Guardian Pro's ability to identify and resolve software defects, including both syntax and logical errors, through its advanced code frame extraction technique, static code analysis, and pattern recognition capabilities. Additionally, the section discusses the impact of project on the software development process, emphasizing its potential to reduce debugging time, enhance development efficiency, and improve software reliability and security. Comparative analysis with traditional bug detection instruments highlights the proactive approach of Code Guardian Pro in error resolution and its capacity to provide developers with technical knowledge for error fixing. Moreover, the section explores the tool's versatility in detecting security risks at deep code layers, across various digital entities like websites and emails, empowering developers to swiftly address vulnerabilities. It also delves into practical implications and potential areas for further development, offering insights into projects role in achieving software reliability, security, and efficiency.

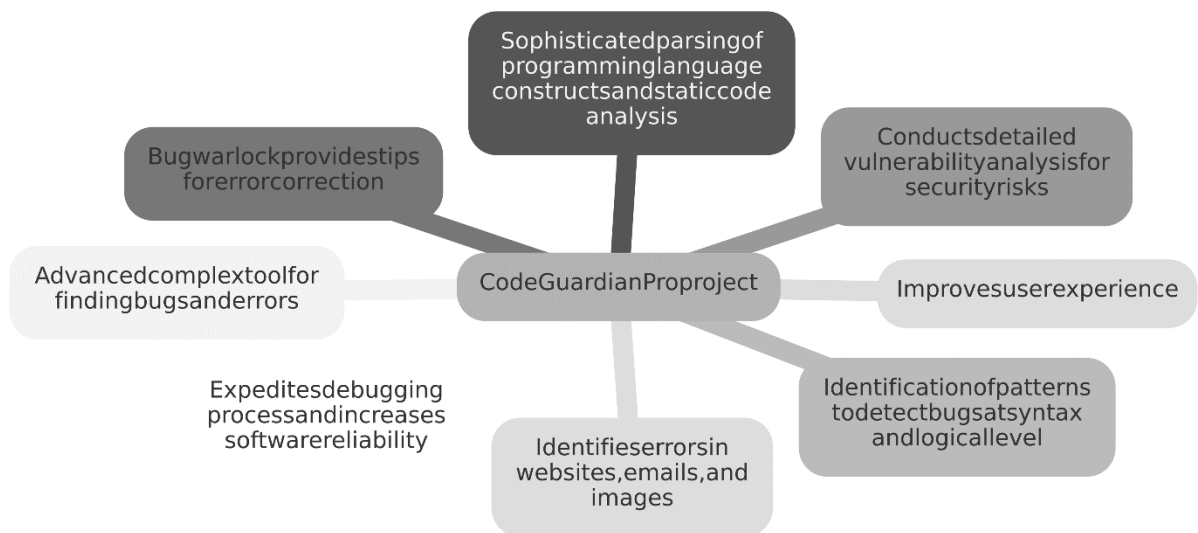


Figure 3.1: Objective diagram of Proposed Method

3.2 Methods Used for Proposed Solution

Python Programming Language: Python was selected as the primary programming language for its versatility, readability, and extensive ecosystem of libraries and frameworks. It is well-suited for tasks such as code analysis, API integration, and data processing.

Bash Scripting: Bash scripting was utilized for system-level operations, file manipulation, and automation of command-line tasks. It provides a convenient way to execute shell commands and interact with the underlying operating system.

Code Frame Extraction Technique: Modern parsing algorithms implemented in Python were used to extract meaningful structures from source code files. These algorithms analyze the code to identify functions, classes, and control flow statements, facilitating further analysis and manipulation.

Static Code Analysis: Python scripts were developed to perform static code analysis, examining source code files for syntactical correctness, adherence to coding standards, and potential vulnerabilities. This analysis helps identify errors and improve code quality before runtime.

VirusTotal API Integration : Integration with the VirusTotal API was achieved using Python, enabling the detection and analysis of potential phishing threats. Python scripts submitted URLs to the VirusTotal platform for scanning and received reports on indicators of compromise, malicious behavior, and reputation scores associated with the submitted URLs.

Exploit Detection and Server Identification: Python scripts were utilized for conducting vulnerability scans to detect potential exploits, while Bash scripting was employed to determine server information, such as the type of server running on port 80, using techniques such as port scanning and banner grabbing.

Syntax Error Checking : Python scripts specifically designed for syntax error checking were used to analyze code snippets and identify syntactical errors. These scripts parsed the code and flagged syntax violations such as missing punctuation, incorrect indentation, or invalid syntax.

Logical Error Verification: Python scripts were developed to identify logical errors in code, ensuring the accuracy of program logic and expected outcomes. These scripts analyzed code logic, conditional statements, and loop constructs to detect potential discrepancies or inconsistencies that may lead to incorrect program behavior.

SQL Injection Detection: Python scripts were employed to conduct a thorough analysis of web applications to identify potential SQL injection vulnerabilities. These scripts analyzed input validation mechanisms, SQL query construction, and database interaction patterns to detect points of vulnerability.

Personal Information Discovery: Python and Bash scripts, along with third-party tools, were utilized to discover personal information associated with phone numbers, email addresses, and IP addresses. These scripts and tools queried public databases, social media platforms, and online resources to retrieve information linked to specific digital entities.

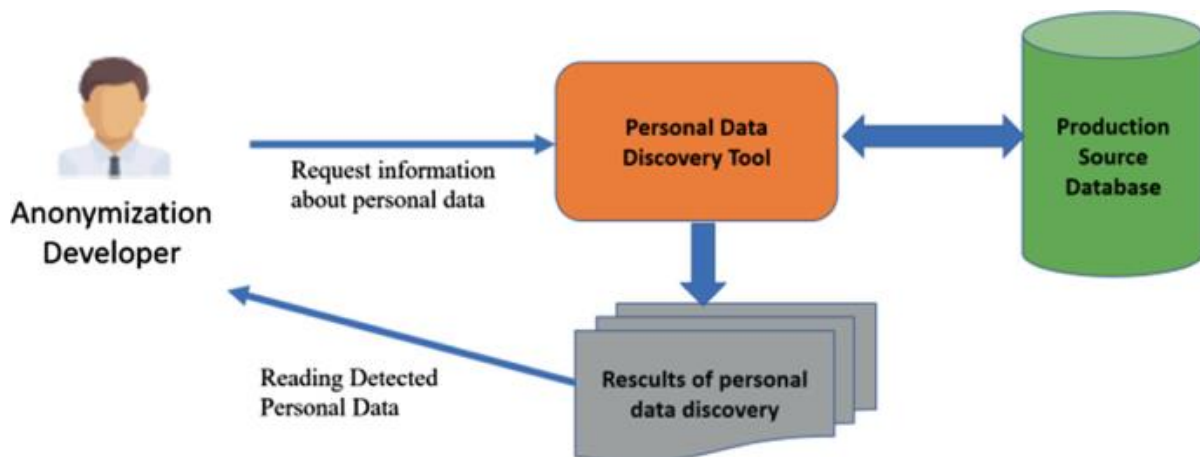


Figure 3.2: Pictorial Representation of Personal Information Discovery

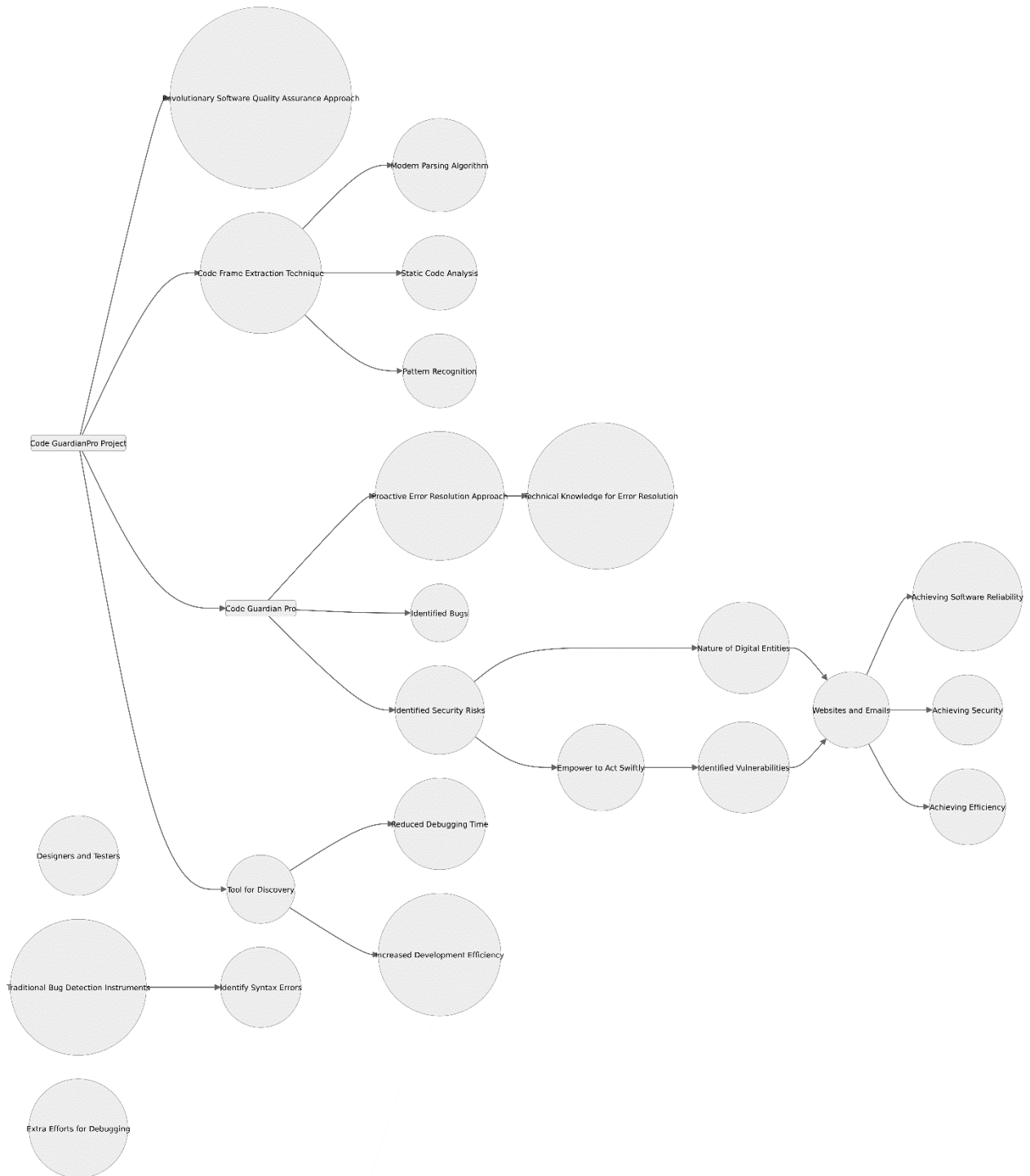


Figure 3.3: Working of Code Guardian Pro Proposed Method

3.3 Designing

3.3.1 UML Diagrams

A. Use Case Diagram: The use case diagram for the project depicts the various interactions between actors and the system, showcasing how different users and entities utilize the system's functionalities. In the diagram, actors such as developers, testers, and administrators are represented, each interacting with the code guardian pro system to perform specific tasks. These tasks include bug detection, error resolution, security analysis, and overall software quality assurance. The UML diagram within this use case representation further illustrates the relationships between these actors and the system's functionalities, providing a visual guide to how users navigate through the features of the Code Guardian Pro tool. With clear delineation of user roles and system functionalities, the use case diagram serves as a valuable tool for understanding the overarching workflow and interactions within the project.

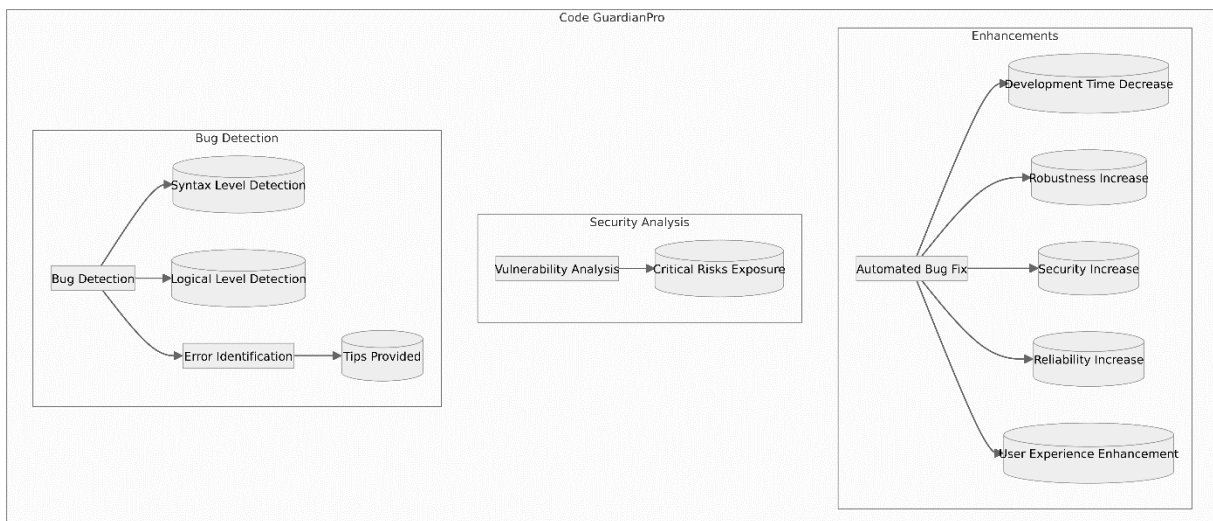


Figure 3.4: Use Case Diagram

B. Sequence Diagram: The sequence diagram for the project illustrates the interactions and flow of messages between its various components during the bug detection and resolution process. At its core, the diagram showcases the seamless integration of functionalities such as code parsing, static analysis, and bug identification within the Code Guardian Pro Tool. The sequence diagram begins with the initiation of bug detection, where the tool invokes the Parsing Algorithm to extract code frames and patterns from the source code. Subsequently, the Bug Warlock component is activated to analyze these patterns and identify any errors or vulnerabilities. Once bugs are detected, the tool provides corrective suggestions to developers, facilitating efficient error resolution. Throughout this process, the Code Inspection and Security Analysis Tools continuously assess the codebase for potential security risks, ensuring comprehensive vulnerability analysis. Overall, the sequence diagram offers a visual representation of the systematic workflow within the Code Guardian Pro project, emphasizing its proactive approach to bug detection and resolution.

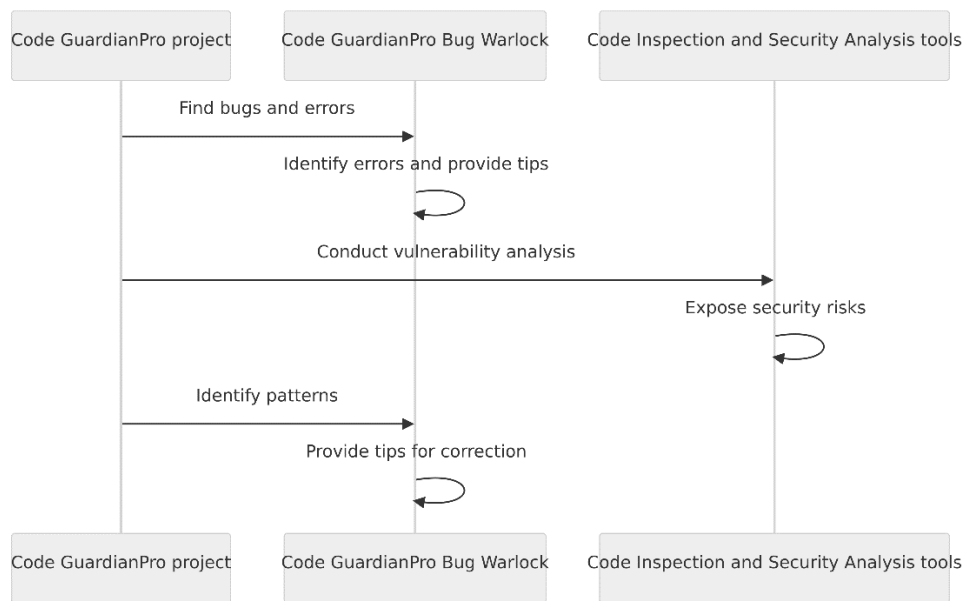


Figure 3.5: Data Flow Diagram

C. Directory Diagram: The directory diagram of the Code Guardian Pro project provides a visual representation of the project's structure and organization. It illustrates the hierarchical arrangement of directories, subdirectories, and files within the project, offering a clear overview of its components and dependencies. Each directory represents a specific module or functionality of the project, while subdirectories further categorize related files and resources. The directory diagram serves as a roadmap for developers, facilitating navigation and understanding of the project's architecture. By delineating the relationships between different components, it aids in efficient code management, version control, and collaboration among team members. Additionally, the directory diagram helps identify potential areas for optimization or restructuring, ensuring the project remains scalable and maintainable as it evolves over time. Overall, the directory diagram plays a crucial role in ensuring the organization, clarity, and maintainability of the project.

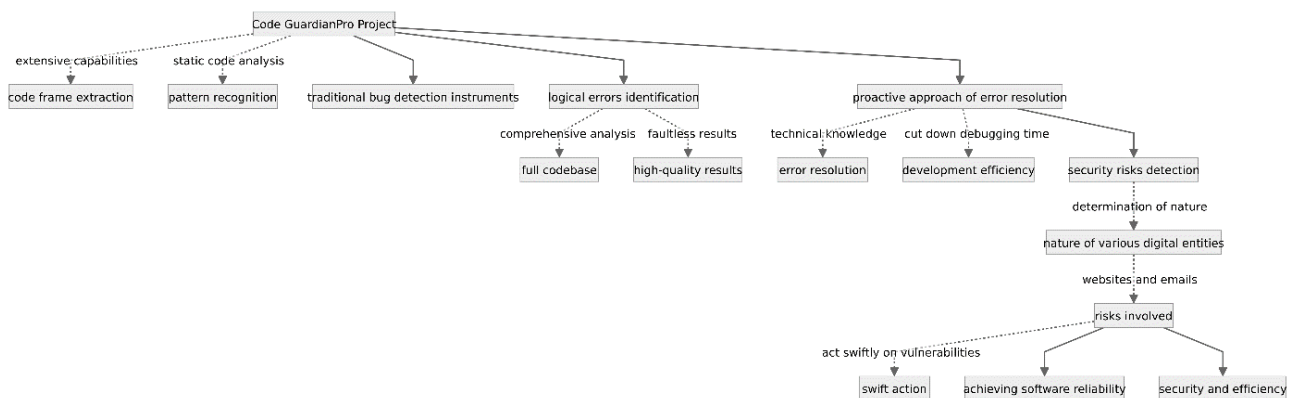


Figure 3.6: Directory Diagram

- Here the sequence diagrams offer clear visualization of interactions; use case diagrams define system functionalities; directory diagrams aid project organization. However, sequence diagrams can become complex, use case diagrams oversimplify, and directory diagrams may become outdated.

3.4 Stepwise Implementation and Code

3.4.1 Implementation steps

To implement the functionalities described in the project in a Linux environment using Python and Bash scripting, we can follow these steps:

A. Finding the Phish Tool:

1. Parser Script (Python)

- Parse the given URL to extract domain and route information.
- Use libraries like ``urllib.parse`` for parsing.

2. Total Wires API Integration (Python):

- Send the parsed URL to the Total Wires API.
- Receive analysis results including key terms.
- Use Python's ``requests`` library for making API calls.

3. File Scanning (Bash):

- Utilize antivirus engines with ``clamscan`` or similar tools.
- Execute scans on files downloaded from the URL.

4. Behavior Analysis (Python):

- Implement dynamic analysis using sandbox environments.
- Monitor for malicious activities.
- Tools like Cuckoo Sandbox can be integrated for this purpose.

5. Threat Detection and Reporting (Python):

- Compile scan data and behavior analysis results.
- Display results and confidence levels.
- Provide analysis prompts for the user.
- Generate downloadable reports.

B. Scan Vulnerability:

1. CMS Detection (Python):

- Utilize tools like CMSmap for CMS detection.
- Extract target web application technology.

2. Information Gathering (Python):

- Collect underlying information about the target website.
- Use tools like WhatWeb for this purpose.

3. Subdomain Gathering (Python):

- Gather and list all relevant subdomains.
- Tools like Sublist3r can be helpful.

4. Multi-threading (Bash/Python):

- Implement parallel processing for faster evaluation.
- Bash scripts with `&` or Python's `concurrent.futures`.

5. Vulnerability Checks (Bash/Python):

- Check for known security issues using tools like Nikto or OWASP ZAP.

6. Auto Shell Injector (Bash/Python):

- Detect and exploit vulnerabilities.
- Implement scripts to inject malicious code.

7. High-level Port Scan (Bash/Python):

- Discover open ports using tools like Nmap.
- Identify potential entry points for hackers.

8. DNS Server Dump (Bash/Python):

- Duplicate DNS server data connected with the target website.
- Utilize tools like dnsrecon.

9. Input Multiple Targets (Python):

- Allow assessment of multiple web applications simultaneously.
- Handle different locations in one scan.

C. Check Syntax Error:

1. Pattern Matching (Python):

- Train a model on a dataset for pattern recognition.
- Identify common syntax errors like missing punctuation or mismatched brackets.

2. Rule-Based Checking (Python):

- Implement rules and constraints of programming languages.
- Enforce compliance with language syntax.

D. Check Logical Error:

1. Data Flow Analysis (Python):

- Analyze the flow of data across the code.
- Detect issues like using variables before defining them.

2. Type Checking (Python):

- Perform type checking to ensure consistency.
- Prevent assigning wrong values to variables.

3. Testing and Debugging (Python):

- Use testing frameworks like unittest for automated testing.
- Debug code using breakpoints and tracing.

E. Find SQL Injection:

1. URL Generator (Python):

- Generate URLs for Wayback Machine's CDX API resource.

2. HTTP Requesting (Python):

- Send HTTP requests to crawl internal URLs.

3. SQL Injection Scanning (Python):

- Identify parameters in webpage address URLs.
- Execute SQL commands from predefined payloads.
- Look for SQL injection vulnerabilities in responses.

F. Find the Person:

1. IP Address Lookup (Bash/Python):

- Utilize IP lookup services or databases.
- Extract geolocation data associated with IP addresses.

2. Gmail Data Extraction (Python):

- Interact with Google APIs to extract data points from Gmail accounts.

3. Phone Number Lookup (Bash/Python):

- Integrate API for reverse phone number lookup.
- Retrieve associated name and spam score.

These are the basic steps and technologies involved in implementing the functionalities described. You would need to write Python scripts for most of the functionalities, while utilizing Bash for executing certain commands or scripts. Additionally, we might need to use various libraries and tools available in Python and Linux environment to achieve specific tasks.

3.4.2 Code

```
#!/bin/bash
B0="$(printf '\033[100m')" S0="$(printf '\033[30m')"
B1="$(printf '\033[91m')" S1="$(printf '\033[31m')"
B2="$(printf '\033[92m')" S2="$(printf '\033[32m')"
B3="$(printf '\033[93m')" S3="$(printf '\033[33m')"
B4="$(printf '\033[94m')" S4="$(printf '\033[34m')"
B5="$(printf '\033[95m')" S5="$(printf '\033[35m')"
B6="$(printf '\033[96m')" S6="$(printf '\033[36m')"
B7="$(printf '\033[97m')" S7="$(printf '\033[37m')"
R1="$(printf '\033[0;1m')" R0="$(printf '\033[00m')"
trap 'printf "\n";rm -rf .torCache;stop;exit 1' 2
stop() {
    torkill
    if [[ $(uname -o) != 'Android' ]]; then
        service tor stop
    fi
}
dependencies() {
    command -v php > /dev/null 2>&1 || { echo >&2 "I require php but it's not installed. Install it.
    Aborting."; exit 1; }
    command -v curl > /dev/null 2>&1 || { echo >&2 "I require curl but it's not installed. Install it.
    Aborting."; exit 1; }
}
menu() {
    echo '      FindMyPhish' | lolcat
    echo '      <x=====>' | lolcat
```



```
elif [[ $Option == 3 || $Option == 03 ]]; then
python3 syn.py file.py
exit
start1
elif [[ $Option == 4 || $Option == 04 ]]; then
flake8 dummy1.py
exit
start1
elif [[ $Option == 5 || $Option == 05 ]]; then
bash vul.sh
exit
start1
elif [[ $Option == 6 ]]; then
clear
bash find.sh
exit 1
elif [[ $Option == 7 ]]; then
bash jai.sh
exit 1
else
printf "\e[1;93m Wrong Option!\e[0m\n"
mpg123 lion.mp3 >/dev/null 2>&1
sleep 1
clear
menu
fi
}
banner() {
```

```
chafa -c 16 --color-space din99d --symbols -dot-stipple logo1.jpg
```

```
echo " ×FindMyPhish
```

```
<×=====×>
```

```
    ×Team 58
```

```
<×=====×>
```

```
" | lolcat
```

```
read -p '$Do you wanna start it: Y/N:' option
```

```
clear
```

```
sleep 2
```

```
echo $B1 'LOADING.....COK' sleep 2
```

```
clear
```

```
echo $B2 '  10%' sleep 0.2
```

```
clear
```

```
echo $B1 '  20%' sleep 0.2
```

```
clear
```

```
echo $B2 '  30%' sleep 0.2
```

```
clear
```

```
echo $B3 '  40%' sleep 0.1
```

```
clear
```

```
echo $B4 '  50%' sleep 0.1
```

```
clear
```

```
echo $B5 '  60%'sleep 0.2
```

```
clear
```

```
echo $B6 '  70%'sleep 0.1
```

```
clear
```

```
echo $B7 '  80%'sleep 0.1
```

```
clear
```

```
echo $B1 '  90%'sleep 0.1
```

```
clear
```

```
echo $B3'  100%'
```

```
printf "\n"
sleep 0.2
clear
}
```

Code for Phishing Tool

```
# Phish Tool
trap 'printf "\n";stop;exit 1' 2
dependencies() {
command -v php > /dev/null 2>&1 || { echo >&2 "I require php but it's not installed. Install it.
Aborting."; exit 1; }
command -v curl > /dev/null 2>&1 || { echo >&2 "I require curl but it's not installed. Install it.
Aborting."; exit 1; }
}
bash red1.sh
echo 'Enter the URL: '
read -p $'>>=>\e[0m\n' username
python3 vtcli.py -u $username
read -p "Do you want to continue using this tool? (y/n): " choice
# Check the user's choice
if [ "$choice" == "y" ] || [ "$choice" == "Y" ]; then
    clear && bash run.sh
elif [ "$choice" == "n" ] || [ "$choice" == "N" ]; then
    # If no, exit the script
    echo "Exiting the script."
    exit 0
else
    # If an invalid choice is entered, display an error message and exit
    echo "Invalid choice. Please enter 'y' or 'n'."
    exit 1
fi
```

Code for Scanning Vulnerability

```
#!/bin/bash

read -p '$Enter the WebSite Name:' opv
cd Sitadel && python3 sitadel.py $opv
# Prompt the user with a yes/no question
read -p "Do you want to continue using this tool? (y/n): " choice
# Check the user's choice
if [ "$choice" == "y" ] || [ "$choice" == "Y" ]; then
    # If yes, execute the script jai.sh
    bash run.sh
elif [ "$choice" == "n" ] || [ "$choice" == "N" ]; then
    # If no, exit the script
    echo "Exiting the script."
    exit 0
else
    # If an invalid choice is entered, display an error message and exit
    echo "Invalid choice. Please enter 'y' or 'n'."
    exit 1
fi
```

Code for finding IP Address

```
#ip addr
#modules required
import argparse
import requests, json
import sys
from sys import argv
import os
```

```
#arguments and parser
parser = argparse.ArgumentParser()
parser.add_argument ("-v", help= "target/host IP address", type=str, dest='target',
required=True )
args = parser.parse_args()
#colours used
red = '\033[31m'
yellow = '\033[93m'
lgreen = '\033[92m'
clear = '\033[0m'
bold = '\033[01m'
cyan = '\033[96m'
#banner of script
print (red+""v 1.0""+red)
ip = args.target
api = "http://ip-api.com/json/"
try:
    data = requests.get(api+ip).json()
    sys.stdout.flush()
    a = lgreen+bold+"[$]"
    b = cyan+bold+"[$]"
    print (a, "[Victim]:", data['query'])
    print(red+"<----->"+red)
    print (b, "[ISP]:", data['isp'])
    print(red+"<----->"+red)
    print (a, "[Organisation]:", data['org'])
    print(red+"<----->"+red)
    print (b, "[City]:", data['city'])
    print(red+"<----->"+red)
```

```
print (a, "[Region]:", data['region'])
    print(red+"<----->" +red)
    print (b, "[Longitude]:", data['lon'])
    print(red+"<----->" +red)
    print (a, "[Latitude]:", data['lat'])
    print(red+"<----->" +red)
    print (b, "[Time zone]:", data['timezone'])
    print(red+"<----->" +red)
    print (a, "[Zip code]:", data['zip'])
    print (" "+yellow)

except KeyboardInterrupt:
    print ('Terminating, Bye'+lgreen)
    sys.exit(0)

except requests.exceptions.ConnectionError as e:
    print (red+"[~]"+" check your internet connection!" +clear)
    sys.exit(1)
```

Code for Email Details

```
#mail
#!/bin/bash
# Prompt the user to enter their email
Read -p "Enter your email: " email
# Run the Python script with the entered email as an argument
cd GHunt && python3 main.py email "$email"
# Prompt the user with a yes/no question
read -p "Do you want to continue using this tool? (y/n): " choice
# Check the user's choice
if [ "$choice" == "y" ] || [ "$choice" == "Y" ]; then
```



```
clear
cd .. && bash find.sh
elif [ "$choice" == "n" ] || [ "$choice" == "N" ]; then
    # If no, exit the script
    echo "Exiting the script."
    exit 0
else
    # If an invalid choice is entered, display an error message and exit
    echo "Invalid choice. Please enter 'y' or 'n'."
    exit 1
fi
```

3.5 System Architecture

The system architecture encompasses a modular design with distinct components for each functionality: URL analysis, vulnerability scanning, syntax and logical error detection, SQL injection detection, and person identification. Each component utilizes APIs or tools tailored to its task, such as Total Wires API for URL analysis and Google APIs for Gmail data extraction. The architecture includes mechanisms for data retrieval, processing, and presentation, with features like multi-threading and asynchronous processing to enhance performance. It incorporates database queries for IP address geolocation and phone number verification, ensuring comprehensive data retrieval. Overall, the architecture emphasizes versatility, scalability, and efficiency in analyzing various types of data for security and identification purposes.

3.5 System Architecture

The system architecture encompasses a modular design with distinct components for each functionality: URL analysis, vulnerability scanning, syntax and logical error detection, SQL injection detection, and person identification. Each component utilizes APIs or tools tailored to its task, such as Total Wires API for URL analysis and Google APIs for Gmail data extraction. The architecture includes mechanisms for data retrieval, processing, and presentation, with features like multi-threading and asynchronous processing to enhance performance. It incorporates database queries for IP address geolocation and phone number verification, ensuring comprehensive data retrieval. Overall, the architecture emphasizes versatility, scalability, and efficiency in analyzing various types of data for security and identification purposes.

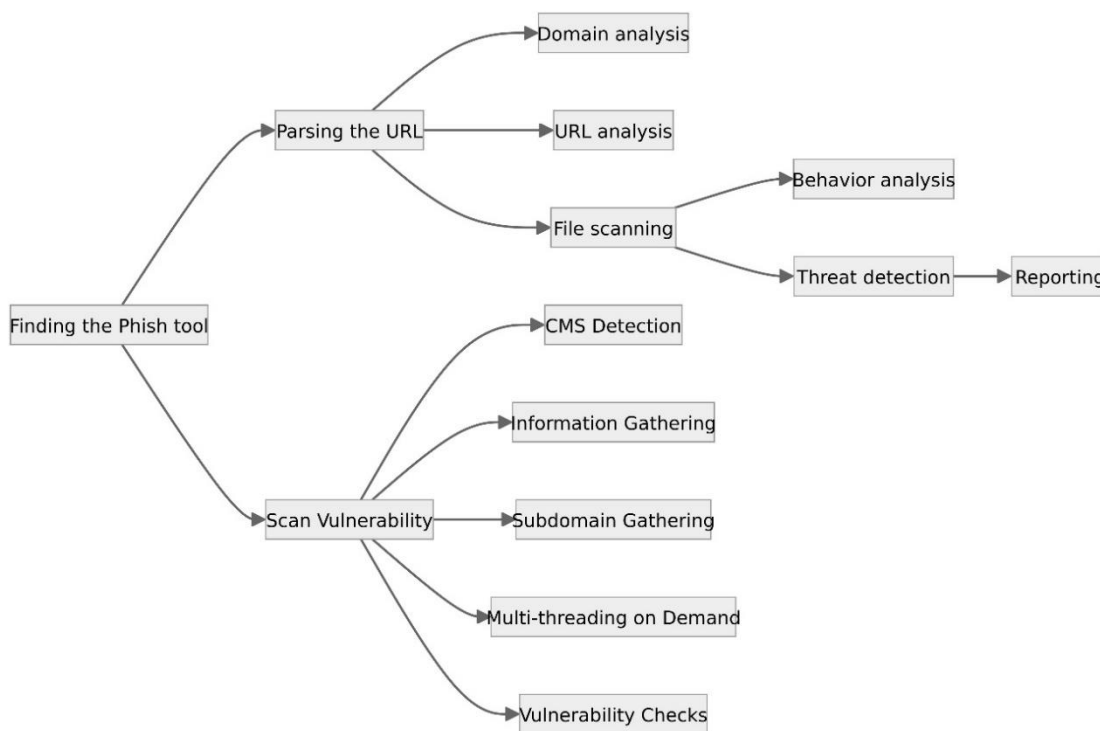


Figure 3.7: System Architecture

This architecture diagram represents the flow of steps, that is how the system will work step by step accordingly.

3.6 System Requirements

3.6.1 Hardware Requirements

- System - Intel(R) Core(TM) i3-7020U CPU
- Speed - 2.30GHz
- Hard Disk - 1 TB.
- Input Devices - Keyboard, Mouse
- Ram - 4 GB.

3.6.2 Software Requirements

- Operating System - Compatible with Windows, macOS, and Linux distributions
- languages - python and bash
- Python Version - Python 3.x (latest version recommended)
- Internet Connectivity - Required for accessing online APIs and databases
- Dependencies - Ensure the installation of necessary Python libraries/modules such as requests, BeautifulSoup, etc., as specified by individual tools within the toolkit.
- Tools - phoneinfoga, ghunt, sqlmap.

CHAPTER 4

RESULTS AND DISCUSSION

CHAPTER 4

RESULTS AND DISCUSSION

The Code Guardian Pro Project represents a paradigm shift in software quality assurance, providing designers and testers with a revolutionary approach. Its advanced capabilities, such as the innovative code frame extraction technique utilizing modern parsing algorithms, static code analysis, and pattern recognition, differentiate it from existing tools. While traditional bug detection tools primarily focus on identifying syntax errors, Code Guardian Pro goes beyond, delving into complexities beyond the syntax level to detect logical errors. This comprehensive methodology involves analyzing the entire codebase, resulting in more accurate and high-quality results.

One of the standout features of project is its proactive approach to error resolution. Unlike many other tools that merely identify issues, the project equips programmers with the technical knowledge needed to fix detected errors. This proactive stance streamlines the debugging process, saving developers valuable time and effort. By providing solutions to identified issues, it accelerates development efficiency.

Moreover, it doesn't stop at bug detection; it also identifies security risks embedded deep within the code. Its versatility extends to various digital entities, including websites and emails, providing developers with a comprehensive understanding of potential risks. This insight enables swift action to address vulnerabilities across multiple platforms.

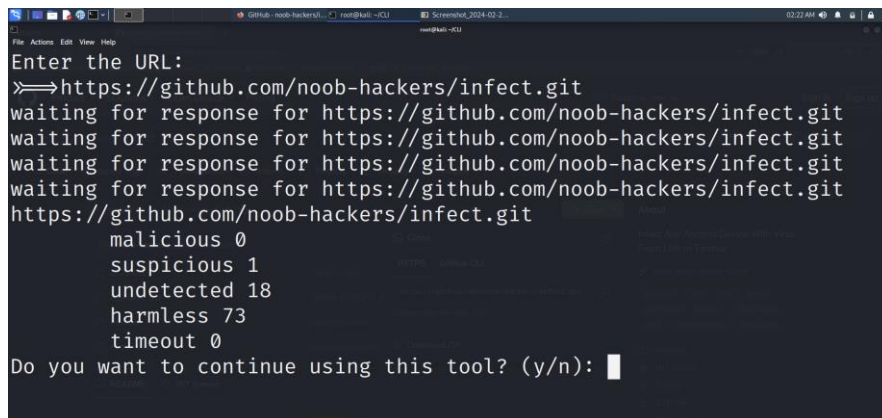
In summary, Code Guardian Pro offers a holistic solution for achieving software reliability, security, and efficiency. With its innovative approach to error detection and resolution, it empowers developers to create robust and secure software applications.

A. Finding the Phish Tool

The first variant will be used build a collector unit which will accept a URL as the input. Then in the back end, we will use total wires API which is an online tool to analyze the URL by comparing factors such comparison of parameters to detect malware, viruses, and any other potential threats. When you do input the desired URL, the tool will be able to generate corresponding key terms.

Follows these steps:

1. **Parsing the URL:** uses the URL the users have given to provide the destination domain and route.
2. **Domain analysis:** it is responsible for identification the domain's reput and any lists of the banned domain names.
3. **URL analysis:** thoroughly scrutinizes the URL's form and content in search for any weird traffic tendency.
5. **File scanning:** makes use of a varying set of antivirus engines to determine if the file may be unsafe by running a check for well-known malware signs.
6. **Behavior analysis:** The alternate option is dynamic analysis, where the file is executed in a sandbox that watches behavior and traces any malicious activities.
7. **Threat detection:** is compiling the scan data from different modules and threat behavior analysis functionality, showing everything in one place in a simple manner and confidence levels.
8. **Reporting:** namely, returns an analysis prompts along with the analysis results which can be downloaded by the user.



```
Enter the URL:
>>>https://github.com/noob-hackers/infect.git
waiting for response for https://github.com/noob-hackers/infect.git
waiting for response for https://github.com/noob-hackers/infect.git
waiting for response for https://github.com/noob-hackers/infect.git
waiting for response for https://github.com/noob-hackers/infect.git
https://github.com/noob-hackers/infect.git
malicious 0
suspicious 1
undetected 18
harmless 73
timeout 0
Do you want to continue using this tool? (y/n):
```

Figure.4.1 Testing Out the Github link for the Phishing or malicious link

B. Scan Vulnerability

The second alternative is about vulnerability scans , it is a tool for penetration testers, bug bounty

hunters or security researchers who have authorization to perform security testing, this tool has multiple exploits that can be used in exploitation of a target and determining whether the target is vulnerable or not. We will also assess how critical the vulnerability is.

1.CMS Detection: The tools are CMS detection tools. Thus, it is possible to identify favorite content management systems such as WordPress, Joomla, PrestaShop, Drupal, OpenCart, Magento or Lokomedia management systems. This allows to be done to discover the basics of the target web application's technology.

2.Information Gathering: It provides mechanism for collecting target information, which can range from the underlying information about target website such as the version, and any publicly available details.

3.Subdomain Gathering: A tool may collect and list all the subdomains pertinent to the target domain as its primary function. This allows increasing the scale of the analysis and finding new weak points of a goal in the target system.

4.Multi-threading on Demand: This tool, with multi-threading, which parallel processes multiple tasks concurrently, has it like this. This way can accelerate the duration of evaluation.

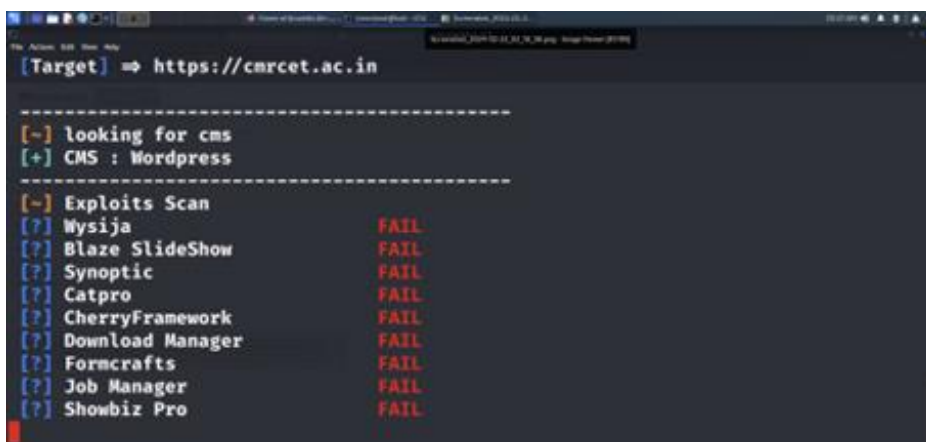
5.Vulnerability Checks: This utility can enhance the process in which target web applications are checked for already known security issues. It is therefore, it is possible to recognize possible failure points that can be used by the attackers as a gateway into an organization's network system.

6.Auto Shell Injector: The tool next to the auto shell injector will detect the vulnerabilities used by attackers to inject malicious code onto various targets' web applications.

7.High-level Port Scan: The tool can use a process of a high-level port scan to locate vulnerabilities in the target web traffic. This can be done to discover open ports which in return, could be used by hackers to steal the data in the targeted infrastructure in an unauthorized way.

8.DNS Server Dump: This tool performs duplication of DNS servers data which are the ones connected with the particulate target website.

9.Input Multiple Targets: The tool allows you to enter shallow as well as deep web applications for assessment. That enables users to see different locations in one scan pause.



```

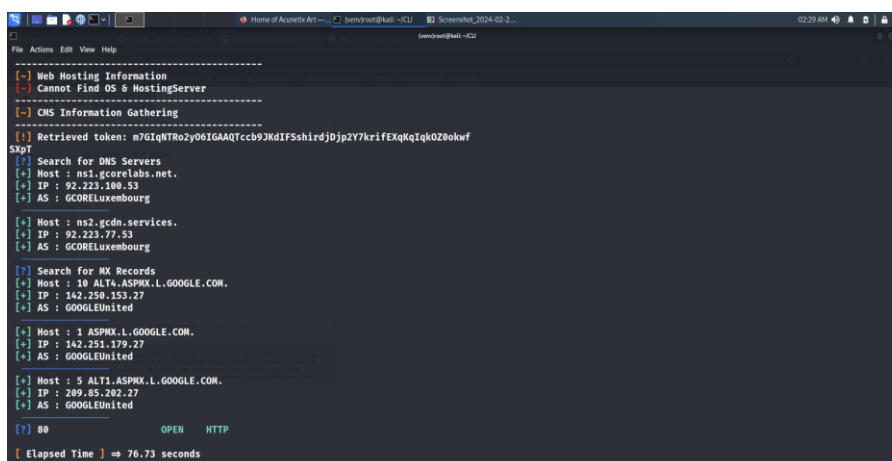
[Target] => https://cmrcet.ac.in

-----
[~] looking for cms
[+] CMS : Wordpress
-----

[~] Exploits Scan
[?] Wysija                                FAIL
[?] Blaze SlideShow                       FAIL
[?] Synoptic                             FAIL
[?] Catpro                               FAIL
[?] CherryFramework                      FAIL
[?] Download Manager                     FAIL
[?] Formcrafts                          FAIL
[?] Job Manager                         FAIL
[?] Showbiz Pro                         FAIL

```

Figure 4.2 Exploits Scan



```

-----
[~] Web Hosting Information
[~] Cannot Find OS & HostingServer
-----

[~] CMS Information Gathering
[+] Retrieved token: m7G1qWTRo2y00IGAAQTcc93KdIFsshirdJp2Y7krifEXqIqk0Z0okuf
-----

[+] Search for DNS Servers
[+] Host : ns1.gcorelabs.net.
[+] IP : 92.223.100.53
[+] AS : GCORELuxembourg

[+] Host : ns2.gcds.services.
[+] IP : 92.223.77.53
[+] AS : GCORELuxembourg

[+] Search for MX Records
[+] Host : 10.ASPMX.L.GOOGLE.COM.
[+] IP : 142.250.153.27
[+] AS : GOOGLEUnited

[+] Host : 1.ASPMX.L.GOOGLE.COM.
[+] IP : 142.251.179.27
[+] AS : GOOGLEUnited

[+] Host : 5.ALT1.ASPMX.L.GOOGLE.COM.
[+] IP : 209.85.202.27
[+] AS : GOOGLEUnited

[+] 88      OPEN  HTTP

[Elapsed Time ] => 76.73 seconds

```

Figure 4.3 Result of Vulnerability Scanning.

C. Check syntax error

Pattern Matching: I have the ability to be trained on a large dataset ranging from the lines of code to vocabulary of natural language which can enable me to identify common patterns and structures associated with valid syntax. Identifiers, which are the parts of the programming code that give instructions to the computer, are a critical element of syntax. An example is a missing punctuation, incorrect keyword, or mismatched bracket may be cues for syntax errors that might need to be corrected.

Rule-Based Checking: Being armed with precise attributes such as rules and constraints of a particular programming language or writing style can be my instrumentality. Compliance with

specific regulations, such as in the case of "cheat" or weak grammar structures, are shown as errors.

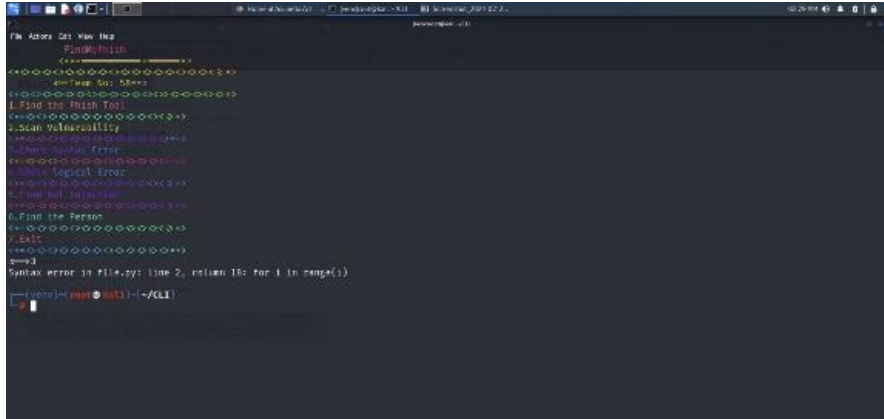


Figure 4.4 Syntax Error

D. Check logical error

Data Flow Analysis: This approach observes the data path across the code, thus locating problems as faulty lines like. This includes using the variables before defining them. Awarding variables their values, but not referring to them later in the program. Duplication of the calculations or the operations that serve no purpose.

Type Checking: Some aspects may carry out type checking for the purpose of maintaining the data type use to be consistent and quality. This will stop problem of assigning wrong values into variables and using wrong operators in respect to certain data types.

Testing and Debugging: Yet, the issue of testing and debugging is not a step of the detection itself; a tool that runs your code and observes its output can find logical errors in there. These tools may offer features like: These tools may offer features like:

- Setting breakpoints to stop at certain points of the run for checking up on the value of the different variables.
- Moving from one code line to another will help you to develop an understanding of the flow of the logic.

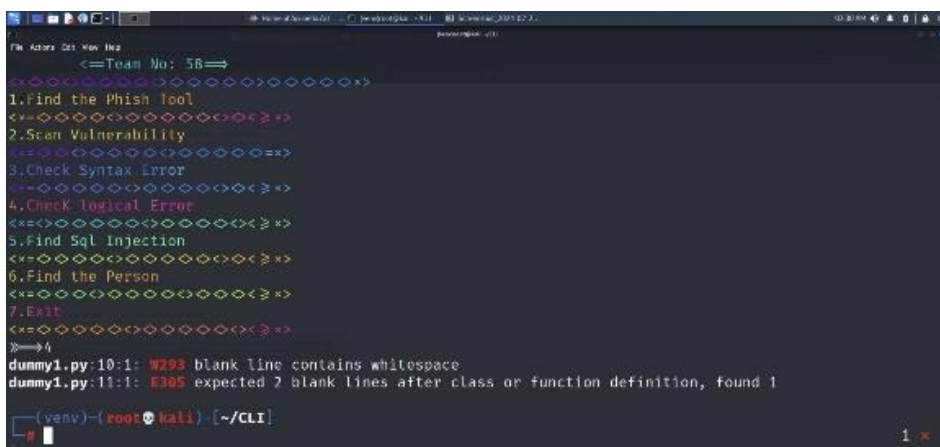


Figure 4.5 Logical Error

E. Find SQL injection

URL Generator: Build URL for downloading Wayback Machine's CDX API resource that the user would provide domain and a subdomain flag.

HTTP Requesting: And extraction functions call the generated URLs to get the responses and the internal URLs are crawled.

SQL Injection Scanning:

- Identifies parameters from the webpage address URLs.
- Executing sql commands from payloads stored in a file.
- Combines URLs and payloads, sending the requests to varying locations.
- Looks for SQL injection granting access vulnerabilities in the response. The script is designed to identify SQL injections against host of URLs received as a parameter while using diverse payloads to execute SQL injections.
- If a response contains some keywords (like "SQL", "Sql") which could mean that a website is vulnerable to SOL injection, it is marked as Threat by some tools.

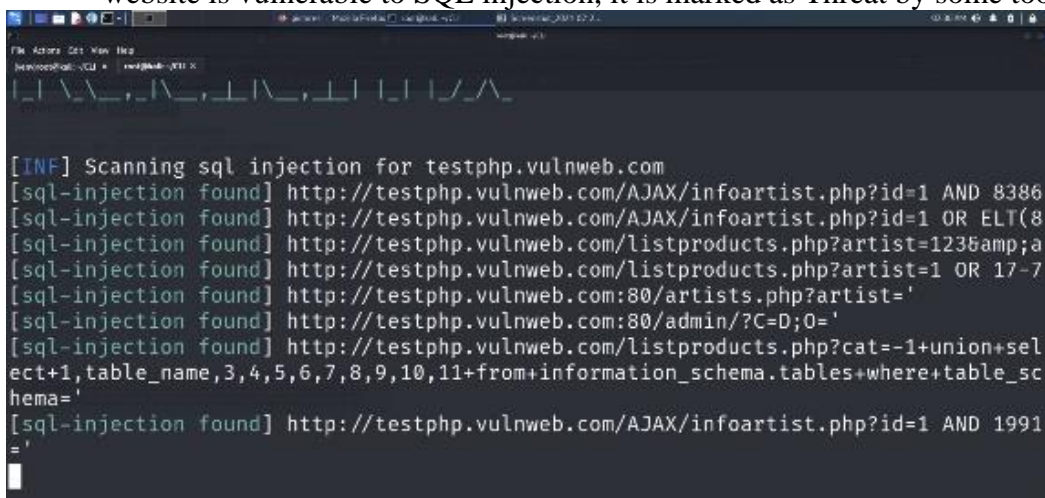


Figure 4.6 SQL Injection Detection

F. Find the person

i. IP Address

An IP lookup works by querying a database or service that contains information about IP addresses and their associated geolocation data and other metadata. Here's a general overview of how an IP lookup works:

IP Address Input: The user provides an IP address as input to the IP lookup tool or service.

Database Query: The IP lookup tool or service sends a query to its internal database or an external database or service that contains information about IP addresses.

Geolocation Data Retrieval: The database or service responds to the query by providing the geolocation data associated with the input IP address. This data typically includes information such as the country, region, city, and even the zip code.

Additional Metadata Retrieval: Depending on the capabilities of the IP lookup tool or service, it may also retrieve additional metadata associated with the input IP address. This can include information such as the name of the internet service provider (ISP), the organization or individual associated with the IP address, and other relevant details.

Data Presentation: The IP lookup tool or service presents the retrieved geolocation data and any additional metadata to the user in a user-friendly format, such as a map displaying the location of the IP address or a table containing the various metadata details.

ii. Gmail

API Interface: Interacts with Google APIs (publicly available interfaces to Google services) to store information.

Data Extraction: Based on the targeted email, GHunt extracts data points from relevant services e.g. Google Account : Name, Google ID YouTube: Channel information (if publicly available)

Google Services: Images, Maps and other functional services (depending on the user's privacy settings).

It uses asynchronous processing to process multiple requests simultaneously, improving performance.

Data Export: Extracted data can be exported in JSON format for further analysis or integration detection. It does not provide the functionality to extract personal information beyond the name associated with the phone number.

iii. Phone number :

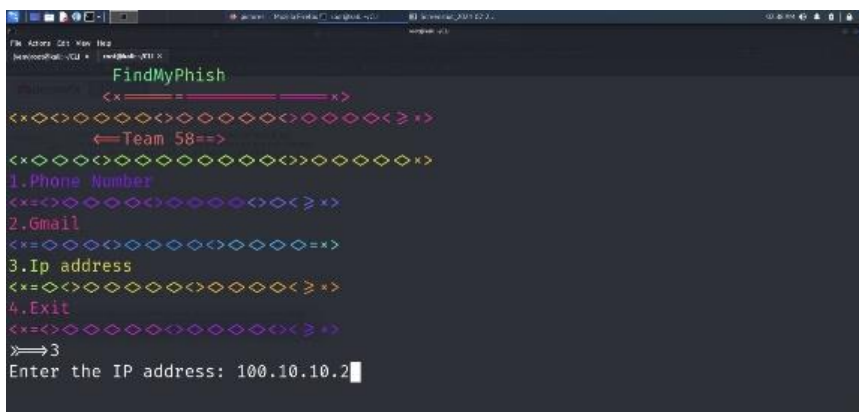
Phone number data searches mainly focus on reverse phone numbers and spam score

Name: The name associated with the phone number (if it exists in the database).

Spam Score: An indicator of the likelihood that a number is spam by a caller.

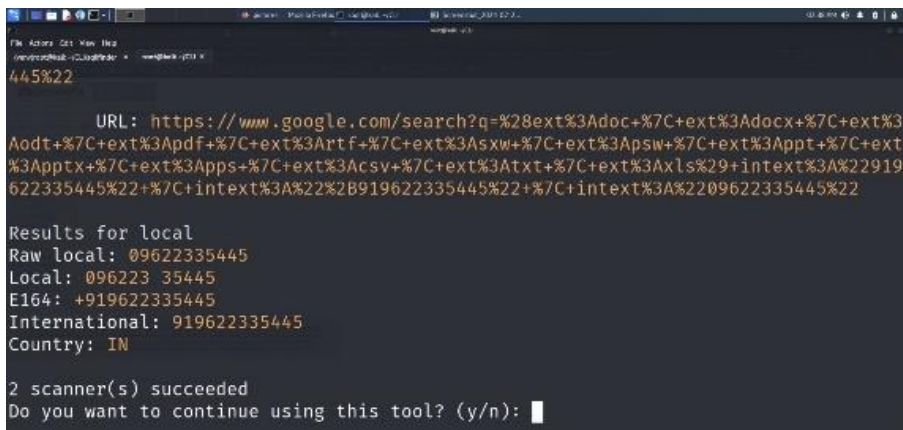
API requests: Developers integrate the API into their applications via authentication keys to make specific requests for phone numbers to be verified.

Data Response: Upon success, the API responds with a structured data structure with name, spam score, and possibly true score (depending on sharing type).



```
FindMyPhish
<=====>
<=====Team 58=====>
1.Phone Number
2.Gmail
3.Ip address
4.Exit
>3
Enter the IP address: 100.10.10.2
```

Figure 4.7 IP Address Lookup



```
445%22
URL: https://www.google.com/search?q=%28ext%3Adoc+%7C+ext%3Adocx+%7C+ext%3Aodt+%7C+ext%3Apdf+%7C+ext%3Artf+%7C+ext%3Asxw+%7C+ext%3Apsw+%7C+ext%3Appt+%7C+ext%3Apptx+%7C+ext%3Apps+%7C+ext%3Acsy+%7C+ext%3Atxt+%7C+ext%3Axls%29+intext%3A%22919622335445%22+%7C+intext%3A%22B919622335445%22+%7C+intext%3A%2209622335445%22
Results for local
Raw local: 09622335445
Local: 096223 35445
E164: +919622335445
International: 919622335445
Country: IN
2 scanner(s) succeeded
Do you want to continue using this tool? (y/n):
```

Figure 4.8 Finding IP Location on Gmaps.

```
File Actions Edit View Help
terminal@kali:~/kali$ ./redpanda.py 100.10.10.2
[$] [Victim]: 100.10.10.2
[$] [ISP]: Verizon Business
[$] [Organisation]: Verizon Business
[$] [City]: Pawtucket
[$] [Region]: RI
[$] [Longitude]: -71.3844
[$] [Latitude]: 41.874
[$] [Time zone]: America/New_York
[$] [Zip code]: 02860
Do you want to continue using this tool? (y/n):
```

Figure 4.9 Phone Number Lookup

```
File Actions Edit View Help
terminal@kali:~/kali$ ./redpanda.py 100.10.10.2
[+] Authenticated !
Google Account data
Name : Twinkle Sharma
[+] Custom profile picture !
=> https://lh3.googleusercontent.com/a-/ALV-UjU6-CrFEjZV2dkh6L3u43p9AYH8X9c11zQj-8rRHoAvAA
[+] Face detected !
[-] Default cover picture
Last profile edit : 2023/12/29 11:55:25 (UTC)
Email : twinklesharna5445@gmail.com
Gaia ID : 116599330150119575999
User types :
- GOOGLE_USER (The user is a Google user.)
Google Chat Extended Data
```

Figure 4. 10 Google Account Data.

```
File Actions Edit View Help
terminal@kali:~/kali$ ./redpanda.py 100.10.10.2
Last profile edit : 2023/12/29 11:55:25 (UTC)
Email : twinklesharna5445@gmail.com
Gaia ID : 116599330150119575999
User types :
- GOOGLE_USER (The user is a Google user.)
Google Chat Extended Data
Entity Type : PERSON
Customer ID : Not found.
Google Plus Extended Data
Enterprise User : False
[+] Activated Google services :
- Maps
Play Games data
```

Figure 4. 11 Gmail Account Data.

CHAPTER 5

CONCLUSION

CHAPTER 5

CONCLUSION

5.1 Conclusion and Future Enhancement

In conclusion, Code Guardian Pro represents a significant advancement in software development, focusing on bug detection and code quality enhancement through innovative approaches such as syntax analysis, static code analysis, and pattern recognition. Its unique versatility extends to various domains including source code, websites, emails, and images, making it robust and comprehensive in its coverage. This adaptability simplifies decision-making for developers by providing intelligence-based evidence, driving development towards innovation and stability. By addressing syntax errors, logical errors, and security vulnerabilities, it ensures software reliability in an era marked by growing digital threats. Moving forward, future enhancements could include further refinement of its analysis techniques, integration of machine learning for more accurate pattern recognition, and expansion of its capabilities to cover emerging technologies and platforms. Ultimately, it sets a new standard in software development, where security is achievable, and user satisfaction is paramount, ushering in a new era of software excellence and reliability.

5.2 Future Work

Looking ahead, the future scope of the project includes integrating with social networks and media platforms to enhance its reach and effectiveness. By incorporating features that allow for seamless interaction with platforms such as Facebook, Twitter, and LinkedIn, Code Guardian Pro can provide developers with insights into potential vulnerabilities arising from social media integration. Additionally, integrating with chatbots like ChatGPT can further augment its capabilities by enabling real-time assistance and feedback during the development process. Moreover, expanding its scope to include analysis of data from social media interactions and chat conversations can provide valuable insights into potential security risks and user experience optimization. Overall, these enhancements will position project as a comprehensive solution for ensuring software reliability, security, and exceptional user experience in an increasingly interconnected digital landscape.

CHAPTER 6

REFERENCES

REFERENCES

- [1] Mansour Alsaleh, Noura Alomar, Monirah Alshreef, Abdulrahman Alari and AbdulMalik Al-Salman, “Performance-Based Comparative Assessment of Open Source Web Vulnerability Scanners”, (2017).
- [2] Sheetal Bairwa, Bhawna Mewara and Jyoti Gajrani, “Vulnerability Scanners: A Proactive Approach to Assess WebApplication Security”, (2014).
- [3] Qiu, S.; Liu, Q.; Zhou, S.; Wu, C. Review of artificial intelligence adversarial attack and defense technologies. *Appl. Sci.* 2019, 9, 909. [Google Scholar] [CrossRef].
- [4] Martins, N.; Cruz, J.M.; Cruz, T.; Abreu, P.H. Adversarial Machine Learning Applied to Intrusion and Malware Scenarios: A Systematic Review. *IEEE Access* 2020, 8, 35403–35419. [Google Scholar] [CrossRef].
- [5] Muslihi, M.T.; Alghazzawi, D. Detecting SQL Injection on Web Application Using Deep Learning Techniques: A Systematic Literature Review. In *Proceedings of the 2020 Third International Conference on Vocational Education and Electrical Engineering (ICVEE)*, Surabaya, Indonesia, 3–4 October 2020. [Google Scholar] [CrossRef].
- [6] Aliero, M.S.; Qureshi, K.N.; Pasha, M.F.; Ghani, I.; Yauri, R.A. Systematic Review Analysis with SQLIA Detection and Prevention Approaches. *Wirel. Pers. Commun.* 2020, 112, 2297–2333. [Google Scholar] [CrossRef].
- [7] Anti-Phishing Working Group (APWG), “Phishing activity trends report— second half 2011,” http://apwg.org/reports/apwg_trends_report_h22011.pdf, 2011, accessed July 2012.
- [8] B. Schneier, “Details of the RSA hack,” http://www.schneier.com/blog/archives/2011/08/details_of_the.html, 2011, accessed December 2011.
- [9] P. Kumaraguru, Y. Rhee, A. Acquisti, L. F. Cranor, J. Hong, and E. Nunge, “Protecting people from phishing: the design and evaluation of an embedded training email system,” in *Proceedings of the SIGCHI conference on Human factors in computing systems*, ser. CHI '07. New York, NY, USA: ACM, 2007, pp. 905–914.
- [10] A. Alnajim and M. Munro, “An anti-phishing approach that uses training intervention for phishing websites detection,” in *Proceedings of the 2009 Sixth International Conference on Information Technology: New Generations*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 405–410.

- [11] S. Gorling, "The Myth of User Education," Proceedings of the 16th Virus Bulletin International Conference, 2006.
- [12] G. Gaffney, "The myth of the stupid user," [12] G. Gaffney, "The myth of the stupid user," <http://www.infodesign.com.au/articles/themythofthestupiduser>, accessed March 2011.
- [13] A. Stone, "Natural-language processing for intrusion detection," Computer, vol. 40, no. 12, pp. 103–105, dec. 2007.
- [14] S. Abu-Nimeh, D. Nappa, X. Wang, and S. Nair, "A comparison of machine learning techniques for phishing detection," in Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit, ser. eCrime '07. New York, NY, USA: ACM, 2007, pp. 60–69.
- [15] C. Yue and H. Wang, "Anti-phishing in offense and defense," in Computer Security Applications Conference, 2008. ACSAC 2008. Annual, 8-12 2008, pp. 345–354.
- [16] P. Knickerbocker, D. Yu, and J. Li, "Humboldt: A distributed phishing disruption system," in eCrime Researchers Summit, 2009, pp. 1–12.
- [17] L. James, Phishing Exposed. Syngress Publishing, 2005.
- [18] J. S. Downs, M. Holbrook, and L. F. Cranor, "Behavioral response to phishing risk," in Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit, ser. eCrime '07. New York, NY, USA: ACM, 2007, pp. 37–44.
- [19] H. Huang, J. Tan, and L. Liu, "Countermeasure techniques for deceptive phishing attack," in International Conference on New Trends in Information and Service Science, 2009. NISS '09, 2009, pp. 636–641.
- [20] T. Moore and R. Clayton, "Examining the impact of website take-down on phishing," in eCrime '07: Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit. New York, NY, USA: ACM, 2007, pp. 1–13.
- [21] S. Egelman, L. F. Cranor, and J. Hong, "You've been warned: an empirical study of the effectiveness of web browser phishing warnings," in Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems, ser. CHI '08. New York, NY, USA: ACM, 2008, pp. 1065–1074.
- [22] M. Wu, R. C. Miller, and S. L. Garfinkel, "Do security toolbars actually prevent phishing attacks?" in Proceedings of the SIGCHI conference on Human Factors in computing systems, ser. CHI '06, New York, NY, USA, 2006, pp. 601–610.

- [23] S. Sheng, B. Wardman, G. Warner, L. F. Cranor, J. Hong, and C. Zhang, “An empirical analysis of phishing blacklists,” in Proceedings of the 6th Conference in Email and Anti-Spam, ser. CEAS’09, Mountain view, CA, July 2009.
- [24] Google, “Google safe browsing API,” <http://code.google.com/apis/safebrowsing/>, accessed Oct 2011.
- [25] Google, “Protocolv2Spec,” <http://code.google.com/p/google-safe-browsing/wiki/Protocolv2Spec>, accessed Oct 2011.
- [26] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta, “Phishnet: predictive blacklisting to detect phishing attacks,” in INFOCOM’10: Proceedings of the 29th conference on Information communications. Piscataway, NJ, USA: IEEE Press, 2010, pp. 346–350.
- [27] Y. Cao, W. Han, and Y. Le, “Anti-phishing based on automated individual white-list,” in DIM ’08: Proceedings of the 4th ACM workshop on Digital identity management. New York, NY, USA: ACM, 2008, pp. 51–60.
- [28] N. Chou, R. Ledesma, Y. Teraguchi, and J. C. Mitchell, “Client-side defense against web-based identity theft,” in NDSS. The Internet Society, 2004.
- [29] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling, “Measuring and Detecting Fast-Flux Service Networks,” in Proceedings of the Network and Distributed System Security Symposium (NDSS), 2008.
- [30] P. LIKARISH, D. DUNBAR, AND T. E. HANSEN, “PHISHGUARD: A browser plug-in for protection from phishing,” in 2nd International Conference on Internet Multimedia Services Architecture and Applications, 2008. IMSAA 2008, 2008, pp. 1 – 6.
- [31] D. L. Cook, V. K. Gurbani, and M. Daniluk, “Phishwish: A stateless phishing filter using minimal rules,” in Financial Cryptography and Data Security, G. Tsudik, Ed. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 182–186.

CHAPTER 7

APPENDIX

Github Link: <https://github.com/hacker795/codeguardianpro>

Paper Link : <https://www.ijert.org/code-guardian-pro>

Published by :
<http://www.ijert.org>

International Journal of Engineering Research & Technology (IJERT)

ISSN: 2278-0181

Volume 13, Issue 03 March 2024

Code Guardian Pro

Mr. V Narasimha, Gopu Chitra Bhanu Reddy, Twinkle Sharma, Jakkidi Santhosh,
Assistant Professor, Department of CSE, CMR College of Engineering & Technology, Hyderabad, Telangana
UG Student, Department of CSE, CMR College of Engineering & Technology, Hyderabad, Telangana

Abstract – The Code Guardian Pro project is an advanced complex tool find bugs and errors in their code as well as the software built and later improved the overall quality and the level of productivity. The tool employs sophisticated parsing of programming language constructs, static code analysis, and identification of patterns to detect bugs both at a syntax level and logical level. The bug warlock in Code Guardian Pro gives more than standard identify errors but also provides briefly but understandable tips on how to correct the identified errors. In addition to this, the Code Inspection and Security Analysis tools are also able to conduct detailed vulnerability analysis, exposing critical security risks found within the source code. Not only it goes through the source codes, but it can also identify devious patterns in websites, emails and even may detect errors in the images. On the other hand, the Code Guardian Pro naturally expedites the debugging, decreases the development time, and increases the robustness, security, and reliability of software applications regardless of the size of the potential bugs and issues. Consequently, the end product is a better user experience.

Keywords – vulnerability scanning, debugging ,logical error, syntax error, files scanning, urls scanning.

I. INTRODUCTION

Developing quality software is essential in today's technology-driven world. However, bugs, errors and weaknesses in the source code can be severe participate in software development. Traditional bug detection methods are often time-consuming and may not detect all possible issues, leading to sub-flat software quality and security risks. To meet this challenge, an improved tool is needed which just works. The process of identifying and fixing bugs, syntax errors, logical problems, and security vulnerabilities in source code, websites, and emails. It is multifaceted software development aimed to deal with exceptional bug exception. Correction and code quality upgrade. It uses both static code analysis and pattern recognizing machines. Algorithms to detect syntax and logical problems, that will fail to expose all the runtime flaws. On the other hand, the project is also about delivering a thorough threat analysis to bring to light possible security loopholes. Risks in the codebase. It applies to programming codes, website content, message formats, and picture quality. Reviewing for full security checking, verifications for thoroughness, and testing. The main aim of this tool is to offer. make the development process shorter by delivering useful directives and simple debugging which, then, brings the time needed for this process to an end and the reforms can ensure the reduction of costs and the rise of quality and security of software development. It stands because information should be a practice of any

developer's kit. Powered by Code Guardian Pro, this automated tool can have positive results but also false positives and false negatives from time to time. necessitating careful developer scrutiny. The primary emphasis here is on static code analysis, of course with the focus on possible

Noticing the lack of runtime errors and the difficulty in dealing with intricate code including unusual methods. While as it only provides the guidance, human expertise still occupies the central position to make complex decisions. Compatibility barriers, demand, infringement on privacy and slowness of learning are the aspects to take into account. Consistently focus on reduced costs, and wise use of licenses should not be a problem for the large development teams alongside the small teams therefore a security audit may need a more detailed retesting process.

II. LITERATURE REVIEW

2.1 NMAP

Nmap [1] is a port scanner which is frequently used to make the list of ports. IP address and all information related to it are caught by chat in it. If an IP is provided, then that tunes find the host sub-domain to which it has been assigned to. It also finds out what interfaces in this host the services reside, how many ports are being run, number of opened ports, closed ports, software or hardware services that are presented by these ports, which may be using TCP, FSP or any other communication protocol. It is able to forecast the type of operating system player is connected to the network. The particular machine host topology is portrayed in the form of a tree diagram whereby the various gateways that are being utilized for localization of the machine are depicted. The next requisite is to open the ports so the attacker will gain an opportunity to steal the intellectual property. A number of ranges of ports can be scanned with a tool such as Nmap.

2.2 NESSUS

Additionally, deploying Nessus [2] discovers the vulnerabilities on the remote host and returns the list of found errors. The stem cell has two types of scan: internal and external. An Origin Scan is an inner scanning depend on the router that is present in the host computer that IP. Other than startion the process identifies the host out of a circules t a certain router. Application web tests which can be performed using a scanner constitute another type of test. Here there are three methods which can be employed; the first one being scanning at the initial instance or the facilitation of a template to be able to perform scanning on an attacked host. One particular machine can perform a plurality of scans in parallel. Nessus is able to make analyzation of vulnerability to the security context, and formulate this context in four typologies which are high, medium,

IJERTV13IS030147

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

low, and information. Results are saved as soon as the host is scanned. Such archival occur either the scanning completed completely for all hosts. Vulnerabilities are disclosed in both ways-vulnerabilities. The report that will be generated can directly go and be used to identify and solve the vulnerability. The former deals with the problems that are listed on the provided form, host compliance inspection and assessment takes care of these accordingly. Results can be exported in any form, e.g. CSV or JSON. Nessus is based on a combination of client-server architecture; it uses HTTP for monitoring, Nessus Responder scanning, and the Nessus Web User Interface. Each session is served by the client, and the server-side validation checks it.

Qabajeh et al. (, 2018) has recently looked at traditional and automated phishing detection by using a very similar methodology. The traditional anti-phishing methods implicate training the individual, holding workshops, and session, the contents of which frequently loop in the legal aspect. In the following passages, Computerized or automated anti-phishing includes list-based and Machine learning-based approaches. The mentioned two techniques are compared giving their similarities, and some elements are positive and others are negative from the user's perspective. Based on the studies, machine learning and rule induction, respectively, are the most appropriate tools that can be used to protect against phishing attacks. The limitations of this work are: the review highlights 67 research items that are the founding principle and the research work does not incorporate Deep Learning methods for online phishing website detection.

Zurairq & Alkasasbeh (2019) conducted an in-depth study of phishing detection methods yielded up to date. The study explicates Anti-Phishing techniques such as Heuristic, Content-Based and Fuzzy rule-based approaches. The finding suggests that phishing websites can be detected more accurately by better methods for identifying them. Research of data was started from 2013 and last till 2018 and that is the base of the project that was carried out. A limitation of this research was the fact that it reviewed only 18 papers; so, more research is required to include Machine Learning, List Based and Deep Learning approaches for phishing website detection.

Benavides et al. (Benavides et al., 2020) conducted a review or an examination of the other researchers' ways to identify phishing (It involved applying the Deep Learning algorithms) in order to enhance the cybersecurity. Finally, the most recognizable rate in the matter of Deep Learning algorithms for fraud email detection is still inadequate. The literature in this work is exclusively based on only 19 studies that were published in 2014 till 2019 journals. Indeed, providing the searched phrase "phishing and Deep Learning" is the way for finding only peer-reviewed articles with the essential topics.

Athulya and Praveen (Athulya, P., 2020) pointed out different phishing attacks that are still a part of the phisher's arsenal, taking into consideration the phisher's most recent phishing tactics as well as some

exploitability by the plugin or exploitability by the host. The detection includes all the vulnerabilities discovered during the scanning and subsequently composed of the CVE lists affected by these anti-phishing strategies. Alongside that, at the heart of the article is a goal of gaining consciousness in the community regarding phishing attacks and methods which have been employed for phishing detection. The investigation stated in the first sentence of this paragraph; clearly the only prevention is to educate users on different types of phishing attacks. The question of choosing what type of security software tool to find a potential phishing email, browser extensions have been found to be the most effective among users. This research was built off of nine articles. The study does not explore Deep Learning methods for vulnerability detection aimed at phishing websites.

Qiu and al. [3] have given a generalized review of ways that artificial intelligence can be used in both attacking and withdrawing in security attacks, primarily at the stages of training and testing. In their work, they aimed at classifying attacks in the classes of natural language processing, cyberspace security, computer vision, and the physical world such attacks. Likewise, they conceptualized the defense sideways in their research and recommended techniques to counter against certain forms of attack. In the work by Martins et al.[4] similar or more than fifteen sample papers were taken that were based on adversarial machine learning techniques used in intrusion detection and malware detection model. In their work, the authors presented the classification of the most common intrusion and malware recognition methods using adversarial attacks and protecting oneself against them.

Muslihi et al. [5] present a review of more than 14 papers that can detect SQL Injection attacks via some deep learning methods which include Carton neural networks, LSTM, DBN, MLP, and Bi-LSTM. Moreover, the authors have analyzed several of the methods aiming at objective, procedure, feature, and dataset points. Muhammad et al. [6] were focused on arranging and evaluating, by means of an analytical approach, the currently known methods and tools that can be applied to prevent an SQL injection attack. The considered studies total 82. The review outcomes revealed that most of the researchers studied and suggested the SQLIA techniques detection approaches (SQLIAs) and not do their best to test the pragmatic approaches.

III. METHODOLOGIES

A. Finding the Phish tool :

The first variant will be used build a collector unit which will accept a URL as the input. Then in the back end, we will use total wires API which is an online tool to analyze the URL by comparing factors such comparison of parameters to detect malware, viruses, and any other potential threats. When you do input the desired URL, the tool will be able to generate corresponding key terms.

Follows these steps:

1. Parsing the URL: uses the URL the users have given to provide the destination domain and route.
2. Domain analysis: it is responsible for identification the domain's reput and any lists of the banned domain names.
3. URL analysis: thoroughly scrutinizes the URL's form and content in search for any weird traffic tendency.
5. File scanning: makes use of a varying set of antivirus engines to determine if the file may be unsafe by running a check for well-known malware signs.
6. Behavior analysis: The alternate option is dynamic analysis, where the file is executed in a sandbox that watches behavior and traces any malicious activities.
7. Threat detection: is compiling the scan data from different modules and threat behavior analysis functionality, showing everything in one place in a simple manner and confidence levels.
8. Reporting: namely, returns an analysis prompts along with the analysis results which can be downloaded by the user.

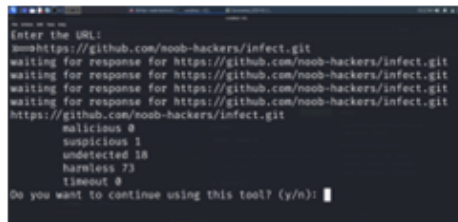


Fig. 1 Testing out the github link for the phishing or malicious link

B. Scan Vulnerability

The second alternative is about vulnerability scans, it is a tool for penetration testers, bug bounty hunters or security researchers who have authorization to perform security testing, this tool has multiple exploits that can be used in exploitation of a target and determining whether the target is vulnerable or not. We will also asses how critical the vulnerability is.

1.CMS Detection: The tools are CMS detection tools. Thus, it is possible to identify favorite content management systems such as WordPress, Joomla, PrestaShop, Drupal, OpenCart, Magento or Lokomedia management systems. This allows to be done to discover the basics of the target web application's technology.

2.Information Gathering: It provides mechanism for collecting target information, which can range from the underlying information about target website such as the version, and any publicly available details.

3.Subdomain Gathering: A tool may collect and list all the subdomains pertinent to the target domain is its

primary function. This allows increasing the scale of the analysis and finding new weak points of a goal in the target system.

4.Multi-threading on Demand: This tool, with multi-threading, which parallel processes multiple tasks concurrently, has it like this. This way can accelerate the duration of evaluation.

5.Vulnerability Checks: This utility can enhance the process in which target web applications are checked for already known security issues. It is therefore, it is possible to recognize possible failure points that can be used by the attackers as a gateway into an organization's network system.

6.Auto Shell Injector: Andthe tool next to the auto shell injector will detect thevulnerabilities used by attackers to inject malicious code onto various targets' web applications.

7.High-level Port Scan: The tool can use a process of a high-level port scan to locate vulnerabilities in the target web traffic. This can be done to discover open ports which in return, could be used by hackers to steal the data in the targeted infrastructure in an unauthorized way.

8.DNS Server Dump: This tool performs duplication of DNS servers data which are the ones connected with the particulate target website.

9.Input Multiple Targets: The tool allows you to enter shallow as well as deep web applications for assessment. That enables users to see different locations in one scan pause.

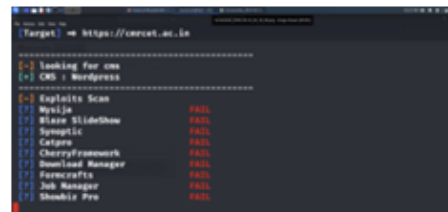


Fig.2 Exploits Scan



Fig.3 Result of vulnerability scanning.

C. Check syntax error

Pattern Matching: I have the ability to be trained on a large dataset ranging from the lines of code to vocabulary of natural language which can enable me to identify common patterns and structures associated with valid syntax. Identifiers, which are the parts of the programming code that give instructions to the computer, are a critical element of syntax. An example is a missing punctuation, incorrect keyword, or mismatched bracket may be cues for syntax errors that might need to be corrected.

Rule-Based Checking: Being armed with precise attributes such as rules and constraints of a particular programming language or writing style can be my instrumentality. Compliance with specific regulations, such as in the case of "cheat" or weak grammar structures, are shown as errors.



Fig.4 syntax error

D. Check logical error

Data Flow Analysis: This approach observes the data path across the code, thus locating problems as faulty lines like. This includes using the variables before defining them, awarding variables their values, but not referring to them later in the program. Duplication of the calculations or the operations that serve no purpose.

Type Checking: Some aspects may carry out type checking for the purpose of maintaining the data type use to be consistent and quality. This will stop problem of assigning wrong values into variables and using wrong operators in respect to certain data types.

Testing and Debugging: Yet, the issue of testing and debugging is not a step of the detection itself; a tool that runs your code and observes its output can find logical errors there. These tools may offer features like, setting breakpoints to stop at certain points of the run to check up on the value of the different variables. Moving from one code line to another will help you to develop an understanding of the flow of the logic.



Fig.5 Logical error

E. Find SQL injection

URL Generator: Build URL for downloading Wayback Machine's CDX API resource that the user would provide domain and a subdomain flag.

HTTP Requesting: And extraction functions call the generated URLs to get the responses and the internal URLs are crawled.

SQL Injection Scanning:

Identifies parameters from the webpage address URLs. Executing sql commands from payloads stored in a file. Combines URLs and payloads, sending the requests to varying locations.

Looks for SQL injection granting access vulnerabilities in the response.

The script is designed to identify SQL injections against host of URLs received as a parameter while using diverse payloads to execute SQL injections.

If a response contains some keywords (like "SQL", "Sql") which could mean that a website is vulnerable to SQL injection, it is marked as Threat by some tools.

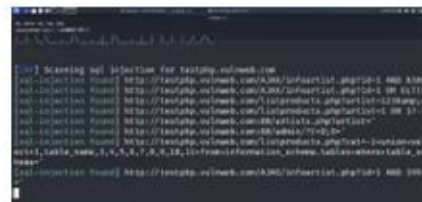


Fig.6 SQL injection detection

F. Find the person

i.IP Address

An IP lookup works by querying a database or service that contains information about IP addresses and their associated geolocation data and other metadata. Here's a general overview of how an IP lookup works:

IP Address Input: The user provides an IP address as input to the IP lookup tool or service.

Database Query: The IP lookup tool or service sends a query to its internal database or an external database or service that contains information about IP addresses.

Geolocation Data Retrieval: The database or service

responds to the query by providing the geolocation data associated with the input IP address. This data typically includes information such as the country, region, city, and even the zip code.

Additional Metadata Retrieval: Depending on the capabilities of the IP lookup tool or service, it may also retrieve additional metadata associated with the input IP address. This can include information such as the name of the internet service provider (ISP), the organization or individual associated with the IP address, and other relevant details.

Data Presentation: The IP lookup tool or service presents the retrieved geolocation data and any additional metadata to the user in a user-friendly format, such as a map displaying the location of the IP address or a table containing the various metadata details.

ii. Gmail

API Interface: Interacts with Google APIs (publicly available interfaces to Google services) to store information.

Data Extraction: Based on the targeted email, GHunt extracts data points from relevant services e.g. Google Account : Name, Google ID YouTube: Channel information (if publicly available)

Google Services: Images, Maps and other functional services (depending on the user's privacy settings). It uses asynchronous processing to process multiple requests simultaneously, improving performance.

Data Export: Extracted data can be exported in JSON format for further analysis or integration.

iii. Phone number :

Phone number data searches mainly focus on reverse phone numbers and spam score detection. It does not provide the functionality to extract personal information beyond the name associated with the phone number.

Name: The name associated with the phone number (if it exists in the database).

Spam Score: An indicator of the likelihood that a number is spam by a caller.

API requests: Developers integrate the API into their applications via authentication keys to make specific requests for phone numbers to be verified.

Data Response: Upon success, the API responds with a structured data structure with name, spam score, and possibly true score (depending on sharing type).



Fig.7 IP address Lookup



Fig.8 Finding IP location on Gmaps.



Fig.9 Phone number lookup



Fig.10 Google account data.



Fig.11 Gmail account data.

IV. RESULTS AND DISCUSSION

The Code GuardianPro Project is a game changer providing designers and testers with a revolutionary software quality assurance approach. Its extensive capabilities, including code frame extraction technique that uses modern parsing algorithm, static code analysis, and pattern recognition, make it the solution that offers something different than other current tools.

The traditional bug detection instruments are mostly used to identify syntax errors. But, Code Guardian Pro is different. It digs into every possibility of a complexity which is even beyond syntax level and identify logical errors as well. This methodology is based on a comprehensive analysis since it involves the full codebase, and the results are more faultless and high-quality.

Code GuardianPro is a standout feature of the debugging system because it is based on the proactive approach of error resolution, which provides programmers with the technical knowledge of how to fix the errors detected. This enables it to outpace the majority of other currently available tools that necessitate extra efforts from developers: from figuring out what has failed, to proposing the solution for the identified issue. Being a tool for discovery rather than dust usage, the project cuts down debugging time. Therefore, it boosts development efficiency. What is more, the tool not only detects and locates devious bugs but also security risks at the deep layer of code. Its versatility in the determination of the nature of various digital entities, including websites and emails, opens the developers sights to a fuller picture of the risks involved. Moreover, it empowers them to act swiftly on the identified vulnerabilities prevalent not only in one but several mediums. Code Guardian Pro offers help in achieving software reliability, security, and efficiency in one package.

V. CONCLUSION

In conclusion, the Code GuardianPro understands the programming as a huge development step in the software developing process. It concentrates on bug-detection and code-quality improvement by using novel approaches of syntax analysis, static code analyzer, and pattern finding. The unique nature of microfinance is that it suits to the local needs successfully perform source code, website, email and images scanning, making it robust and covers all possibilities. This adaptability drives development by simplifying decision-making processes for developers, giving them intelligence-based evidence. While processing the digitized sets in this age of growing digital vulnerabilities it make sure the software reliability by detecting syntax, logical errors, and security issues. Code GuardianPro paves the way for a new trend, which is a from repairing to innovation and then to achieving stable and secure software which performs exceptionally and enables great user experience. As a result, this project translates the age of software development when security is both possible and pure satisfaction.

VI. REFERENCES

- [1] Mansour Alsaleh, Noura Alomar, Monirah Alshreef, Abdulrahman Alari and AbdulMalik Al-Salman, "Performance-Based Comparative Assessment of Open Source Web Vulnerability Scanners", (2017).
- [2] Sheetal Bairwa, Bhawna Mewara and Jyoti Gajrani, "Vulnerability Scanners: A Proactive Approach to Assess Web Application Security", (2014)
- [3] Qiu, S.; Liu, Q.; Zhou, S.; Wu, C. Review of artificial intelligence adversarial attack and defense technologies. *Appl. Sci.* **2019**, *9*, 909. [Google Scholar] [CrossRef]

- [4] Martins, N.; Cruz, J.M.; Cruz, T.; Abreu, P.H. Adversarial Machine Learning Applied to Intrusion and Malware Scenarios: A Systematic Review. *IEEE Access* **2020**, *8*, 35403–35419. [Google Scholar] [CrossRef]
- [5] Muslihi, M.T.; Alghazzawi, D. Detecting SQL Injection on Web Application Using Deep Learning Techniques: A Systematic Literature Review. In *Proceedings of the 2020 Third International Conference on Vocational Education and Electrical Engineering (ICVEE)*, Surabaya, Indonesia, 3–4 October 2020. [Google Scholar] [CrossRef]
- [6] Aliero, M.S.; Qureshi, K.N.; Pasha, M.F.; Ghani, I.; Yauri, R.A. Systematic Review Analysis with SQLIA Detection and Prevention Approaches. *Wirel. Pers. Commun.* **2020**, *112*, 2297–2333. [Google Scholar] [CrossRef]
- [7] Anti-Phishing Working Group (APWG), "Phishing activity trends report—second half 2011," http://apwg.org/reports/apwg_trends_report_h22011.pdf, 2011, accessed July 2012.
- [8] B. Schneier, "Details of the RSA hack," http://www.schneier.com/blog/archives/2011/08/details_of_the.html, 2011, accessed December 2011.
- [9] P. Kumaraguru, Y. Rhee, A. Acquisti, L. F. Cranor, J. Hong, and E. Nunge, "Protecting people from phishing: the design and evaluation of an embedded training email system," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, ser. CHI '07. New York, NY, USA: ACM, 2007, pp. 905–914.
- [10] A. Alnajim and M. Munro, "An anti-phishing approach that uses training intervention for phishing websites detection," in *Proceedings of the 2009 Sixth International Conference on Information Technology: New Generations*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 405–410.
- [11] S. Gorling, "The Myth of User Education," *Proceedings of the 16th Virus Bulletin International Conference*, 2006.
- [12] G. Gaffney, "The myth of the stupid user," <http://www.infodesign.com.au/articles/themythofthestupiduser>, accessed March 2011.
- [13] A. Stone, "Natural-language processing for intrusion detection," *Com-puter*, vol. 40, no. 12, pp. 103–105, dec. 2007.
- [14] S. Abu-Nimeh, D. Nappa, X. Wang, and S. Nair, "A comparison of machine learning techniques for phishing detection," in *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, ser. eCrime '07. New York, NY, USA: ACM, 2007, pp. 60–69.
- [15] C. Yue and H. Wang, "Anti-phishing in offense and defense," in *Com-puter Security Applications Conference*, 2008. ACSAC 2008. Annual, 8–12 2008, pp. 345–354.
- [16] P. Knickerbocker, D. Yu, and J. Li, "Humboldt: A distributed phishing disruption system," in *eCrime Researchers Summit*, 2009, pp. 1–12.
- [17] L. James, *Phishing Exposed*. Syngress Publishing, 2005.
- [18] J. S. Downs, M. Holbrook, and L. F. Cranor, "Behavioral response to phishing risk," in *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, ser. eCrime '07. New York, NY, USA: ACM, 2007, pp. 37–44.
- [19] H. Huang, J. Tan, and L. Liu, "Countermeasure techniques for deceptive phishing attack," in *International Conference on New Trends in Information and Service Science*, 2009. NISS '09, 2009, pp. 636–641.
- [20] T. Moore and R. Clayton, "Examining the impact of website take-down phishing," in *eCrime '07: Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*. New York, NY, USA: ACM, 2007, pp. 1–13.
- [21] S. Egelman, L. F. Cranor, and J. Hong, "You've been warned: an empirical study of the effectiveness of web browser phishing warnings," in *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, ser. CHI '08. New York, NY, USA: ACM, 2008, pp. 1065–1074.
- [22] M. Wu, R. C. Müller, and S. L. Garfinkel, "Do security toolbars actually prevent phishing attacks?" in *Proceedings of the SIGCHI conference on Human Factors in computing systems*, ser. CHI '06. New York, NY, USA, 2006, pp. 601–610.
- [23] S. Sheng, B. Wardman, G. Warner, L. F. Cranor, J. Hong, and C. Zhang, "An empirical analysis of phishing blacklists," in *Proceedings of the 6th Conference in Email and Anti-Spam*, ser. CEAS'09. Mountain view, CA, July 2009.
- [24] Google, "Google safe browsing API," <http://code.google.com/apis/safebrowsing/>, accessed Oct 2011.
- [25] Google, "Protocol v2Spec," <http://code.google.com/p/google-safe-browsing/wiki/Protocolv2Spec>, accessed Oct 2011.

- [26] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta, "Phishnet: predictive blacklisting to detect phishing attacks," in INFOCOM'10: Proceedings of the 29th conference on Information communications. Piscataway, NJ, USA: IEEE Press, 2010, pp. 346–350.
- [27] Y. Cao, W. Han, and Y. Le, "Anti-phishing based on automated individual white-list," in DEM '08: Proceedings of the 4th ACM workshop on Digital identity management. New York, NY, USA: ACM, 2008, pp. 51–60.
- [28] N. Chou, R. Ledesma, Y. Teraguchi, and J. C. Mitchell, "Client-side defense against web-based identity theft," in NDSS. The Internet Society, 2004.
- [29] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling, "Measuring and Detecting Fast-Flux Service Networks," in Proceedings of the Network and Distributed System Security Symposium (NDSS), 2008.
- [30] P. Likarish, D. Dunbar, and T. E. Hansen, "Phishguard: A browser plug-in for protection from phishing," in 2nd International Conference on Internet Multimedia Services Architecture and Applications, 2008. IMSAA 2008, 2008, pp. 1–6.
- [31] D. L. Cook, V. K. Gurbani, and M. Daniluk, "Phishwish: A stateless phishing filter using minimal rules," in Financial Cryptography and Data Security, G. Tsudik, Ed. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 182–186.
- [32] Y. Zhang, J. I. Hong, and L. F. Cranor, "Cantina: a content-based approach to detecting phishing web sites," in Proceedings of the 16th international conference on World Wide Web, ser. WWW '07. New York, NY, USA: ACM, 2007, pp. 639–648.

