VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi-560014

Software Testing

Mini Project Report

On

"RapidScan-Based Software Testing and Validation"

Submitted in partial fulfillment of the requirement of VI Semester

Software Testing Laboratory (21ISL66)

Submitted by

VIJAY KUMAR GOWDA K K

Under the guidance of

Prof. Swathi Darla
Asst. Professor
Dept. of ISE
RVITM, Bangalore.

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

RV INSTITUTE OF TECHNOLOGY AND MANAGEMENT®
Chaitanya Layout, JP Nagar 8th Phase, Kothanur, Bengaluru-560076

2023-2024

RV INSTITUTE OF TECHNOLOGY AND MANAGEMENT, BANGALORE - 560076
(Affiliated to VTU, Belgaum)

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the mini-project work entitled **"RapidScan-Based Software Testing and Validation"** is carried out by **Vijay Kumar Gowda K K** in partial fulfillment for the requirement of VI Semester Software Testing Laboratory (21ISL66) in **Information Science and Engineering** of the **Visvesvaraya Technological University, Belagavi** during the year 2023-2024. It is certified that all the corrections/ suggestions indicated for the given internal assessment have been incorporated in the report. This report has been approved as it satisfies the academic requirements with respect to the mini-project work.

Signature of the Project Coordinator:
Prof. Swathi Darla
Asst. Professor, Dept. of ISE
RVITM, Bangalore

Signature of Head of the Department:
Dr. Latha C A
Prof. & Head, Dept. of ISE
RVITM, Bangalore

External Viva

Name of Examiners	**Signature with date**
1	
2	

ACKNOWLEDGEMENT

We express our profound gratitude to **Dr. Jayapal R, Principal, RVITM, Bangalore**, for providing the necessary facilities and an ambient environment to work.

We are grateful to **Dr. Latha C A, Head of Department, Information Science and Engineering, RVITM, Bangalore**, for her valuable suggestions and advice throughout my/our work period.

We would like to express our deepest gratitude and sincere thanks to our project coordinator, **Dr. Shruthi P** and lab coordinator **Prof. Girish Kumar B C**, for their keen interest and encouragement in the project, whose guidance made the project into reality.

We would like to thank all the staff members of the Department of Information Science and Engineering for their support and encouragement during the course of this project.

ABSTRACT

RapidScan is adept at identifying a wide array of security issues that commonly plague web applications. Among the critical vulnerabilities it targets are SQL injection, cross-site scripting (XSS), and various forms of server misconfigurations. SQL injection attacks exploit vulnerabilities in the database layer, potentially allowing attackers to manipulate database queries and access sensitive information. XSS vulnerabilities enable attackers to inject malicious scripts into web pages viewed by other users, compromising user data and session integrity. Server misconfigurations, on the other hand, can expose the application to a range of attacks due to improper settings or outdated software. By leveraging multiple specialized scanning tools, RapidScan provides extensive coverage across these diverse vulnerability types, ensuring that potential security threats are thoroughly detected and reported.

The multi-tool approach of RapidScan offers significant advantages in terms of coverage and flexibility. Each integrated tool within RapidScan is designed to excel at identifying specific types of vulnerabilities, thereby complementing each other and filling in potential gaps left by any single tool. This comprehensive approach ensures that no significant security threats are overlooked during the scanning process. Additionally, the ability to aggregate results from various tools provides a holistic view of the security landscape, allowing security professionals to prioritize remediation efforts effectively.

Furthermore, RapidScan's modular design allows for the easy addition of new tools and updates, keeping the scanner relevant and effective against emerging threats. This adaptability is crucial in the ever-evolving field of cybersecurity, where new vulnerabilities and attack vectors are continuously discovered. By maintaining an up-to-date arsenal of scanning capabilities, RapidScan ensures that organizations can proactively defend against the latest threats. The combination of extensive coverage, flexibility, and adaptability makes RapidScan an indispensable tool for comprehensive vulnerability assessment and robust web application security.

LIST OF FIGURES

TABLE OF CONTENTS

Chapter 1

Introduction

1.1 Objectives of this Testing

The primary objectives of testing RapidScan are multi-faceted, encompassing several key aspects of vulnerability assessment:

Automation of Vulnerability Scanning

RapidScan [1] aims to automate the vulnerability scanning process by integrating a variety of open-source tools. This automation helps to:

- Reduce manual effort and time required for conducting comprehensive scans.
- Increase the consistency and reliability of scan results.
- Enable frequent and regular vulnerability assessments without significant resource allocation.

Centralized Vulnerability Assessment Platform

By providing a centralized platform, RapidScan aims to:

- Consolidate scan results from multiple tools into a single, unified report.
- Provide an easy-to-use interface for configuring and executing scans.
- Facilitate easier analysis and interpretation of scan results by security professionals.

Simplification of Setup and Execution

RapidScan [1] strives to simplify the setup and execution of vulnerability scans through:

- A modular and user-friendly interface.
- Easy integration of additional scanning tools.
- Streamlined installation processes for various environments (e.g., Ubuntu, Docker).

1.2 Scope of this Testing

The scope of RapidScan's testing encompasses a wide range of security aspects, including but not limited to:

Network Vulnerabilities

RapidScan is designed to identify and analyze network vulnerabilities such as:

- Open ports and services.
- MisconFigd network devices.
- Weaknesses in network protocols.

Web Application Vulnerabilities

RapidScan targets common web application vulnerabilities, including:

- SQL Injection (SQLi).
- Cross-Site Scripting (XSS).
- Cross-Site Request Forgery (CSRF).
- Broken Authentication and Session Management.

System Misconfigurations

RapidScan [1] helps in detecting system misconfigurations that can lead to security breaches, such as:

- Default or weak passwords.
- Improperly conFigd security settings.
- Outdated software and unpatched vulnerabilities.

Sensitive Information Exposure

The tool scans for potential exposures of sensitive information, such as:

- Unencrypted sensitive data.
- Disclosure of sensitive information through error messages.
- Improperly secured data storage.

Chapter 2

Requirement Analysis

2.1 Hardware Requirements

To effectively run RapidScan [1], certain hardware specifications are recommended to ensure optimal performance and reliability. These requirements include:

Minimum Hardware Specifications

- **RAM:** 4GB or higher to handle the memory requirements of multiple scanning tools running concurrently.
- **CPU:** Dual-core processor to efficiently manage parallel processes.
- **Storage:** At least 10GB of free disk space to store scan results and temporary files.
- **Network:** Stable internet connection for downloading tool updates and accessing online resources.

Recommended Hardware Specifications

- **RAM:** 8GB or higher for improved performance, especially when dealing with large-scale scans.
- **CPU:** Quad-core processor to enhance the tool's ability to handle multiple tasks simultaneously.
- **Storage:** 20GB or more to accommodate extensive scan data and logs.
- **Network:** High-speed internet connection to ensure quick download and upload speeds.

2.2 Software Requirements

RapidScan relies on a specific set of software requirements to ensure compatibility and functionality. These include:

Operating System

- **Ubuntu:** Version 18.04 or later is recommended due to its stability and compatibility with a wide range of open-source tools.

Essential Software Packages

- **Python:** Version 3.6 or later is required as RapidScan is built using Python.
- **Docker:** For containerized setups, Docker should be installed to manage and run the RapidScan container.

Python Libraries

- **Requests:** For making HTTP requests to various web services.
- **Nmap:** Python bindings for the Nmap [2] tool to integrate network scanning capabilities.
- **Other Dependencies:** Specified in the requirements.txt file within the RapidScan [1] repository.

2.3 Tool Used

2.3.1 Features of RapidScan

RapidScan is equipped with several powerful features designed to enhance its functionality and usability:

Multi-Tool Integration

- **Integration of Over 80 Scanning Tools:** RapidScan supports a wide array of scanning tools, each specialized in different types of vulnerability detection.
- **Automatic Updates:** The tool can automatically update the integrated scanning tools to ensure the latest vulnerabilities and techniques are covered [3].

Automation and Scheduling

- **Automated Scanning:** Users can automate the entire scanning process, reducing the need for manual intervention.
- **Scheduled Scans:** The ability to schedule scans at regular intervals ensures continuous monitoring and assessment.

Comprehensive Reporting

- **Detailed Reports:** RapidScan generates comprehensive reports that include detailed information about detected vulnerabilities, their severity, and remediation steps.
- **Customizable Reports:** Users can customize the report format to suit their specific needs and preferences.

Modular Design

- **Ease of Integration:** The modular design of RapidScan [1] allows for easy integration of additional scanning tools and functionalities.
- **Flexibility:** Users can enable or disable specific modules based on their requirements.

User-Friendly Interface

- **Command-Line Interface (CLI):** The CLI provides a simple yet powerful interface for configuring and executing scans.
- **Configuration Files:** Users can use configuration files to specify scan parameters and settings, making it easy to replicate and automate scans as shown in Fig 2.1 .

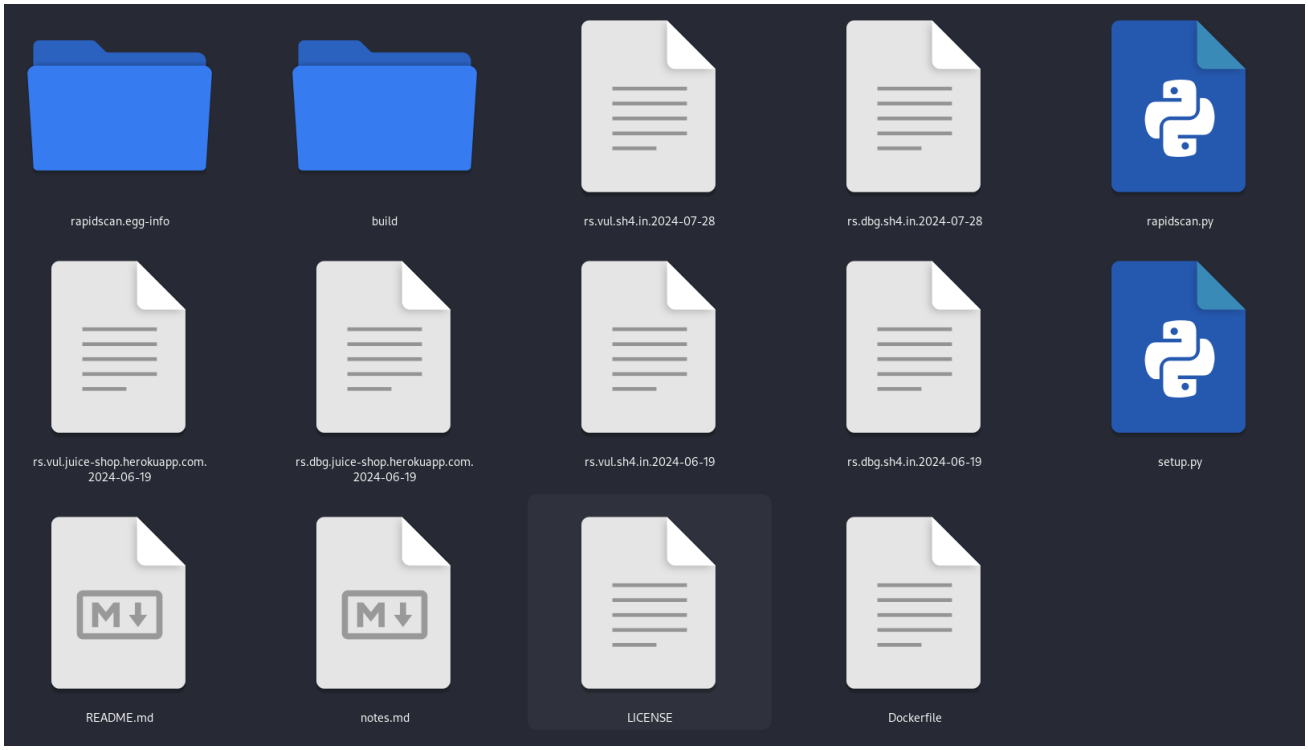

Fig 2.1: RapidScan File Architecture Diagram

The above Fig 2.1 illustrates the file architecture of RapidScan, highlighting the organization and interaction of various components and scripts within the tool's directory structure.

2.3.2 Architecture of RapidScan

RapidScan's architecture is designed to be flexible, scalable, and easy to extend. It comprises several key components:

Central Controller (rapidscan.py)

The central controller script (rapidscan.py) is the core of RapidScan. It orchestrates the scanning process by:

- **Initialization:** Setting up the scanning environment and loading configuration files.
- **Tool Execution:** Initiating the execution of integrated scanning tools and managing their outputs.
- **Result Aggregation:** Collecting and aggregating scan results from multiple tools.
- **Report Generation:** Compiling the aggregated results into a detailed report.

Integration Modules

Each scanning tool is integrated into RapidScan through dedicated modules. These modules handle:

- **Tool Execution:** Running the scanning tool with specified parameters.
- **Output Parsing:** Parsing the output generated by the tool to extract relevant information.
- **Result Formatting:** Formatting the parsed output into a consistent format for aggregation.

Results Aggregation and Reporting Module

This module is responsible for:

- **Aggregating Results:** Combining results from different scanning tools into a single, unified dataset.
- **Normalizing Data:** Ensuring consistency in the format and structure of aggregated data.
- **Generating Reports:** Creating detailed reports that highlight detected vulnerabilities, their severity, and recommended remediation steps.

Fig 2.2: Module Interaction in RapidScan

The above Fig 2.2 depicts the interaction between different modules in RapidScan, showing how the main script coordinates with integrated scanning tools to perform comprehensive vulnerability assessments.

2.3.3 Main Script

The rapidscan.py script serves as the central controller for the tool. Its responsibilities include:

Initialization

- **Configuration Loading:** Loading configuration settings from files or command-line arguments.
- **Environment Setup:** Preparing the environment for running scans, including setting up necessary directories and files.

Execution

- **Tool Management:** Managing the execution of integrated scanning tools.
- **Parallel Execution:** Running multiple tools in parallel to improve scan efficiency.

Result Collection

- **Output Handling:** Collecting output from each tool and storing it in temporary files.
- **Error Handling:** Managing errors and exceptions that occur during the scanning process.

Report Generation

- **Data Aggregation:** Aggregating results from all tools into a single dataset.
- **Report Formatting:** Formatting the aggregated data into a comprehensive report.
- **Output Delivery:** Saving the report to a specified location or sending it to a specified recipient.

python

```
def start_scans(self):
    for tool in self.tools:
        tool.run()
```

python

```
def collect_results(self):
    results = []
    for tool in self.tools:
        results.append(tool.get_results())
    return results
```

python

```
def generate_report(self, results):
    report = "Vulnerability Assessment Report\n"
    for result in results:
        report += result.to_string() + "\n"
    return report
```

2.4 Setup and Installation

Setting up RapidScan involves installing necessary software packages, configuring the environment, and running the tool. The setup process varies slightly depending on the platform (Ubuntu or Docker).

2.4.1 Ubuntu

To install RapidScan on Ubuntu, follow these steps:

Clone Repository
First, clone the RapidScan repository from GitHub:

```
git clone https://github.com/skavngr/rapidscan.git
cd rapidscan
```

Install Dependencies
Update the package list and install Python 3 and pip:

```
sudo apt-get update
sudo apt-get install python3 python3-pip
```
Install required Python libraries:

```
pip3 install -r requirements.txt
```

Run RapidScan
Finally, execute the rapidscan.py script to start the tool:

```
python3 rapidscan.py
```
Note: Ensure that you have the necessary permissions to run the script and access required resources.

2.4.2 Docker

Using Docker for RapidScan offers a convenient and isolated environment, ensuring consistency across different systems. To set up RapidScan with Docker, follow these steps:

Build Docker Image

Build the Docker image using the provided Dockerfile:

docker build -t rapidscan .

Run Docker Container

Run the Docker container to start RapidScan:

docker run -it rapidscan

Note: Ensure that Docker is installed and properly conFigd on your system.

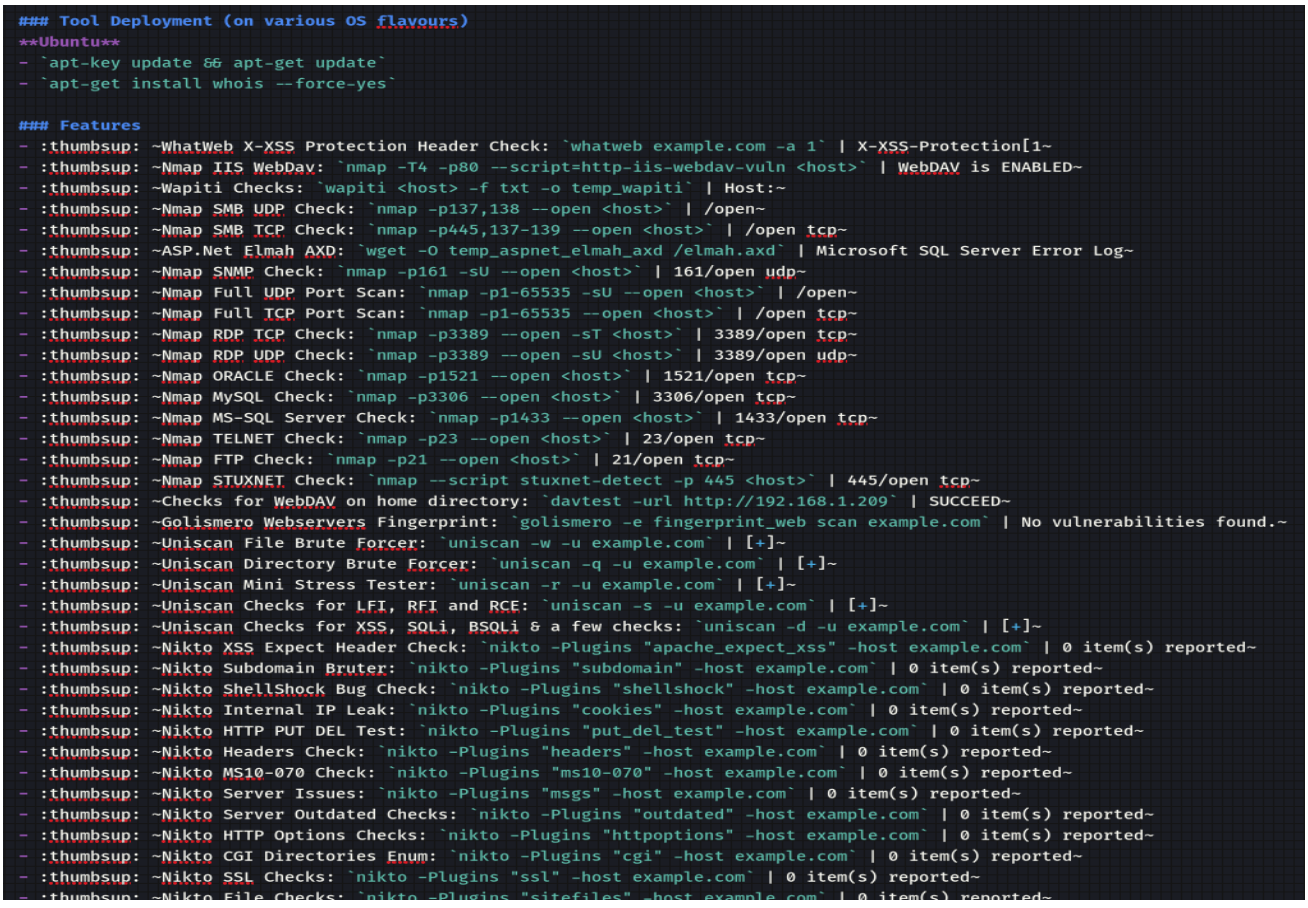

Fig 2.3: Deployment Features of RapidScan

The Fig 2.3 potrays deployment features of RapidScan are shown here, emphasizing the tool's adaptability and ease of deployment across different environments, including local systems and Docker containers.

Chapter 3

Methods of Testing

Testing RapidScan [1] involves various methodologies to ensure its effectiveness, reliability, and performance in real-world scenarios. These methods include practical applications, tool integration, execution, and result compilation.

3.1 Practical Applications

RapidScan is utilized in multiple practical applications across different domains. Some of the key applications include:

Enterprise Network Security

RapidScan is employed by organizations to conduct regular vulnerability assessments of their network infrastructure. This includes scanning for:

- Open ports and services that may expose the network to attacks.
- MisconFigd network devices that could lead to unauthorized access.
- Weaknesses in network protocols that may be exploited by attackers.

Web Application Security

Web application security is a critical concern for organizations that rely on web-based services. RapidScan is used to:

- Identify common web application vulnerabilities such as SQL Injection (SQLi), Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF).
- Assess the security of authentication and session management mechanisms.
- Detect insecure direct object references and security misconfigurations.

Compliance Testing

Organizations are often required to comply with industry standards and regulations such as PCI DSS, HIPAA, and GDPR. RapidScan assists in:

- Conducting regular vulnerability assessments to ensure compliance with security standards.
- Generating detailed reports that can be used as evidence of compliance.
- Identifying and addressing non-compliance issues before regulatory audits.

3.2 Integration of Scanning Tools

RapidScan integrates a wide range of scanning tools to provide comprehensive vulnerability assessments. Each tool is integrated through dedicated modules that handle its execution and result processing. For example:

Nmap Integration

Nmap is a popular network scanning tool used to discover hosts and services on a network. RapidScan integrates Nmap [2] using a dedicated module:

python

```
from nmap import PortScanner

class NmapScanner:
    def __init__(self):
        self.scanner = PortScanner()

    def run(self):
        self.scanner.scan(hosts='localhost', arguments='-T4 -A')

    def get_results(self):
        return self.scanner.csv()
```

SQLMap Integration

SQLMap [3] is an open-source tool used to detect and exploit SQL injection vulnerabilities. RapidScan integrates SQLMap as follows:

python

```
import subprocess

class SQLMapScanner:
    def __init__(self, target_url):
        self.target_url = target_url

    def run(self):
        subprocess.run(['sqlmap', '-u', self.target_url, '--batch'])

    def get_results(self):
        # Parse SQLMap output for results
        pass
```

3.3 Execution and Interpretation of Results

Executing RapidScan involves running the rapidscan.py script, which initiates the scanning process and manages the execution of integrated tools. The steps include:

Starting RapidScan

To start RapidScan, use the following command:

sh

```
python3 rapidscan.py
```

Scan Execution

RapidScan executes each integrated tool according to the specified configuration. The tools run in parallel, allowing for efficient and comprehensive scanning.

Collecting Results

The results from each tool are collected and stored in temporary files. RapidScan then parses these results to extract relevant information about detected vulnerabilities.

Interpreting Results

The results are aggregated and normalized to ensure consistency. The final report includes:

- **Vulnerability List:** Detailed information about each detected vulnerability.
- **Severity Ratings:** Classification of vulnerabilities based on their severity (e.g., critical, high, medium, low).
- **Remediation Recommendations:** Suggested actions to address each vulnerability.

Chapter 4

Test Cases Developed

Developing test cases for each module of RapidScan ensures that the tool functions correctly and reliably. The test cases cover both functional and non-functional aspects of the tool.

4.1 Functional Testing

Functional testing focuses on verifying that each function within the tool performs as expected. The test cases include:

Initialization Function

- **Test Case:** Verify that the initialization function correctly loads configuration settings and sets up the environment.
- **Expected Result:** The configuration settings are loaded, and the environment is prepared without errors.

Tool Execution Function

- **Test Case:** Verify that the tool execution function initiates the execution of all conFigd scanning tools[5].
- **Expected Result:** All conFigd tools are executed, and their outputs are collected.

Result Collection Function

- **Test Case:** Verify that the result collection function correctly collects and parses the output from each tool.
- **Expected Result:** The output from each tool is collected and parsed without errors.

Report Generation Function

- **Test Case:** Verify that the report generation function creates a comprehensive and accurate report.
- **Expected Result:** The report includes detailed information about detected vulnerabilities, their severity, and remediation steps.

4.2 Non-Functional Testing

Non-functional testing assesses the performance, usability, and reliability of RapidScan. The test cases include:

Performance Testing

- **Test Case:** Assess the tool’s efficiency and speed in handling large scans.
- **Expected Result:** The tool completes scans within a reasonable time frame, even for large datasets.

Usability Testing

- **Test Case:** Ensure that the tool’s interface is user-friendly and intuitive.
- **Expected Result:** Users can easily conFig and execute scans without encountering usability issues[4].

Reliability Testing

- **Test Case:** Confirm the tool's stability under various conditions.
- **Expected Result:** The tool remains stable and performs correctly, even under stress conditions[6].

Chapter 5

Types of Testing Done

Testing RapidScan involves several types of testing to ensure comprehensive coverage and validation of the tool's functionality and performance.

5.1 Unit Testing

Unit testing involves testing individual modules and functions to ensure they work correctly in isolation. The focus is on:

- Verifying the correctness of each function.
- Identifying and fixing bugs at an early stage.
- Ensuring that changes in one part of the code do not affect other parts.

Example Unit Test

python

```
def test_initialize():
    rapidscan = RapidScan()
    assert rapidscan.initialize() == True

def test_execute_tool():
    rapidscan = RapidScan()
    rapidscan.add_tool('nmap')
    assert rapidscan.execute_tool('nmap') == True
```

5.2 Integration Testing

Integration testing focuses on verifying that different components of RapidScan work together seamlessly. The objectives include:

- Ensuring that integrated tools function correctly when executed by RapidScan.

- Verifying the correctness of data exchange between modules.
- Identifying and resolving integration issues.

Example Integration Test

python

```
def test_integration():
    rapidscan = RapidScan()
    rapidscan.add_tool('nmap')
    rapidscan.add_tool('sqlmap')
    assert rapidscan.start_scans() == True
    results = rapidscan.collect_results()
    assert len(results) > 0
```

5.3 System Testing

System testing ensures that RapidScan performs its intended functions when integrated into a complete system. The focus is on:

- Validating the overall functionality of the tool.
- Ensuring that all components work together as expected.
- Identifying and fixing system-level issues.

Example System Test

python

```
def test_system():
    rapidscan = RapidScan()
    rapidscan.add_tool('nmap')
    rapidscan.add_tool('sqlmap')
    rapidscan.start_scans()
    results = rapidscan.collect_results()
    report = rapidscan.generate_report(results)
    assert 'Vulnerability Assessment Report' in report
```

5.4 Acceptance Testing

Acceptance testing validates the tool against the requirements and objectives set forth. The focus is on:

- Ensuring that RapidScan meets the needs of its users.
- Verifying that the tool performs as expected in real-world scenarios.
- Identifying and addressing any gaps between the tool's functionality and user requirements.

Example Acceptance Test

```
def test_acceptance():
    rapidscan = RapidScan()
    assert rapidscan.run() == True
    report = rapidscan.generate_report(rapidscan.collect_results())
    assert 'Critical' in report
    assert 'Remediation' in report
```

5.5 Acceptance of Test Cases

The acceptance of test cases involves validating the effectiveness of the developed test cases. The objectives include:

- Ensuring that test cases cover all critical aspects of the tool's functionality.
- Verifying the accuracy and reliability of test results.
- Identifying and resolving any deficiencies in the test cases.
-

Example Test Case Acceptance

```
def test_case_acceptance():
    test_cases = [
        test_initialize,
        test_execute_tool,
        test_acceptance]
    for test in test_cases:
        assert test() == True
```

Chapter 6

Results

Result Compilation

Result compilation involves aggregating and normalizing the results from multiple scanning tools. The process includes:

Aggregating Results

RapidScan combines results from different tools into a single, unified dataset. This involves merging data from various sources and ensuring consistency in the format and structure.
python

```
def aggregate_results(self, results):
    aggregated_results = {}
    for result in results:
        for vulnerability in result:
            if vulnerability not in aggregated_results:
                aggregated_results[vulnerability] = 0
            aggregated_results[vulnerability] += 1
    return aggregated_results
```

Generating Reports

The final step is to generate a detailed report that presents the aggregated and normalized results in a clear and concise format. The report includes:

- **Executive Summary:** An overview of the assessment and key findings.
- **Detailed Vulnerability Analysis:** In-depth analysis of each detected vulnerability.
- **Remediation Recommendations:** Suggested actions for addressing the identified vulnerabilities.

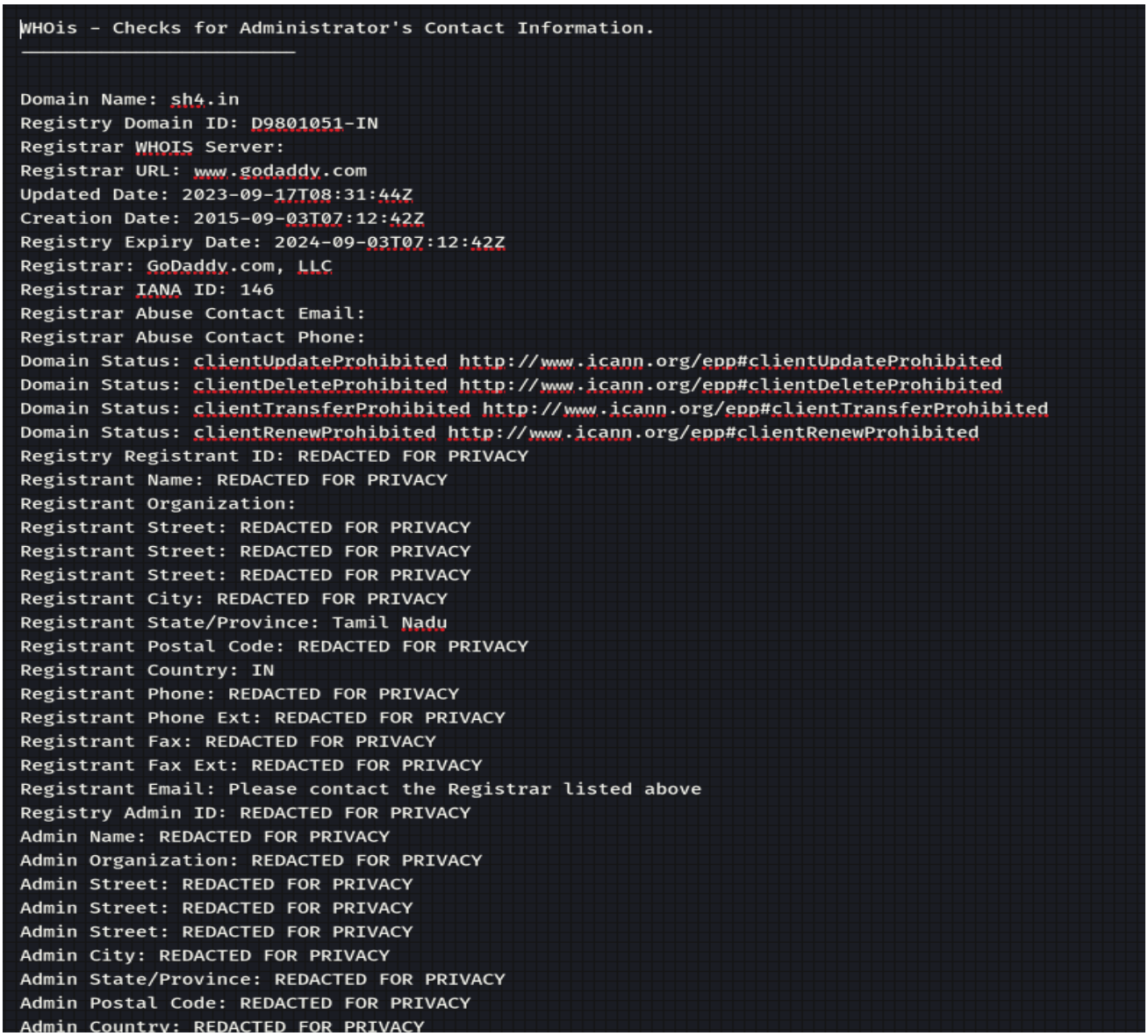

Fig 6.1: Detailed Report Example

The above Fig 6.1 presents an example of a detailed report generated by RapidScan, demonstrating the format and depth of information provided, including vulnerability details and remediation recommendations.

Normalizing Data

Normalization ensures that the data is consistent and can be easily interpreted. This involves standardizing the format of vulnerability descriptions, severity ratings, and remediation steps.

Chapter 7

Conclusion

RapidScan stands out as a robust and comprehensive tool for vulnerability assessment, leveraging the power of multiple integrated open-source scanning tools to provide thorough and automated security analysis. Its modular design and user-friendly interface make it accessible to both novice and experienced security professionals, enabling efficient detection and remediation of a wide range of vulnerabilities. The integration of tools like Nmap and SQLMap [3] ensures that network, web application, and system vulnerabilities are thoroughly assessed, providing organizations with a holistic view of their security posture. The detailed reporting capabilities of RapidScan, which include severity ratings and remediation recommendations, facilitate informed decision-making and prompt action to mitigate identified risks.

Furthermore, the extensive testing methodologies employed in the development of RapidScan, including unit, integration, system, and acceptance testing, ensure that the tool performs reliably in various environments and scenarios. The setup and installation process, whether on Ubuntu or via Docker, is straightforward, enabling quick deployment and utilization. As cyber threats continue to evolve, RapidScan's ability to incorporate updates and new scanning tools positions it as a vital asset for continuous security monitoring and improvement. By automating the vulnerability assessment process, RapidScan not only saves time and resources but also enhances the overall security posture of organizations, making it an indispensable tool in the cybersecurity landscape.

References

[1] S. Khandelwal, "RapidScan: The Multi-Tool Vulnerability Scanner," GitHub Repository. Available: https://github.com/skavngr/rapidscan

[2] "Nmap: The Network Mapper," Nmap.org. Available: https://nmap.org

[3] "SQLMap: Automatic SQL Injection and Database Takeover Tool," sqlmap.org. Available: http://sqlmap.org

[4] "Nikto2: Web Server Scanner," CIRT.net. Available: https://cirt.net/Nikto2

[5] "Burp Suite: Web Vulnerability Scanner," PortSwigger.net. Available: https://portswigger.net/burp

[6] "Metasploit Framework: Penetration Testing Software," Rapid7.com. Available: https://www.metasploit.com