WEBGUARD– ADVANCED WEBSITE VULNERABILITY ANALYSER TOOL

A PROJECT REPORT

Submitted by

MADHAN RAJ R	**-**	**513520104021**
NITISH C	**-**	**513520104025**
THILAK A	**-**	**513520104031**
VIJAYALAYAN K	**-**	**513520104041**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

ANNAI MIRA COLLEGE OF ENGINEERING AND TECHNOLOGY

ARAPAKKAM, RANIPET-632517.

ANNA UNIVERSITY:: CHENNAI 600 025

APRIL/MAY 2024

BONAFIDE CERTIFICATE

Certified that this project report **"WEBGUARD– Advanced Website Vulnerability Analyser Tool"** is the bonafide work of **"MADHAN RAJ R (513520104021), NITISH C (513520104025), THILAK A (513520104039), VIJAYALAYAN K (513520104041)"** who carried out the project work under my supervision.

SIGNATURE

Mr.S.SRINIVASAN M.TECH,(Ph.D),
HEAD OF THE DEPARTMENT

Dept. of Computer Science and Engg,
Annai Mira College of Engineering and Technology, Arapakkam
Ranipet-632517

SIGNATURE

Mrs.J.SAVITHRI M.E,
SUPERVISOR

Dept. of Computer Science and Engg,
Annai Mira College of Engineering and Technology, Arapakkam
Ranipet-632517

Submitted for the project viva voce held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

I first offer my deepest gratitude to Almighty God who has given me strength and good health during the course of this project.

I am conscious about my indebtedness our Honorable Chairman **Mr.S.Ramadoss**, Secretary **Mr.G.Thamothiran** and our Principal **Dr.T.K.Gopinathan, Ph.D.**, and our Vice Principal **Dr.D.Saravanan, Ph.D.**, who inspired me reach greater heights in the pursuits of knowledge.

We proudly express our esteemed gratitude to our Head of the Department **Mr.S.SRINIVASAN, M.TECH,(Ph.D)** for his encouragement and assistance towards the completion of the project.

Special thanks must be mentioned to our internal guide **Mrs.J.SAVITHRI M.E, Assistant Professor**, Computer Science Department for her expert advice, valuable information and guidance throughout the completion of the project. We also express our sincere thanks to our project coordinator, guide and all our staff of CSE Department who helped us in the completion of this project.

We also express our sincere thanks to our project coordinator, guide and all our staff of CSE Department who helped us in the completion of this project.

TABLE OF CONTENT

WEBGUARD– ADVANCED WEBSITE VULNERABILITY ANALYSER TOOL

ABSTRACT

WebGuard represents a sophisticated solution in the realm of cybersecurity, offering an advanced website vulnerability analysis tool designed to fortify digital defenses against evolving threats. By amalgamating cutting-edge network scanning methodologies with meticulous security checks, WebGuard scrutinizes target websites for a spectrum of common vulnerabilities, including open ports, outdated software, weak authentication protocols, sql injection, cross site scripting CSRF and absent security headers. Through its user-friendly interface, users can seamlessly input target URLs or IP addresses, with the option to specify additional ports for a comprehensive scan.

The tool's robust scanning process not only identifies vulnerabilities but also provides actionable recommendations to mitigate risks and enhance overall security posture. Emphasizing usability and efficacy, WebGuard empowers users with insights into potential weaknesses and equips them with the tools needed to proactively address vulnerabilities before they can be exploited by malicious actors.

In a landscape increasingly characterized by digital threats, WebGuard stands as a beacon of defense, offering a proactive approach to cybersecurity. By facilitating informed decision-making and enabling proactive security measures, WebGuard aims to bolster the resilience of web infrastructure and safeguard digital assets against a myriad of cyber threats. This abstract encapsulates the essence of WebGuard's mission and functionality, positioning it as a crucial component in the arsenal of organizations seeking to safeguard their online presence.

LIST OF FIGURES

LIST OF ABBREVIATIONS

URL : Uniform Resource Locator

IP : Internet Protocol

HTTPS : Hypertext Transfer Protocol Secure

SFTP : SSH File Transfer Protocol

GUI : Graphical User Interface

XSS : Cross-Site Scripting

SSL : Secure Sockets Layer

INTRODUCTION

CHAPTER 1

1. INTRODUTION

1.1. PROBLEM STATEMENT

In today's digital landscape, website security is of paramount importance due to the increasing frequency and sophistication of cyber threats. Despite advancements in cybersecurity measures, websites remain vulnerable to various forms of attacks such as unauthorized access, data breaches, and service disruptions. Traditional security approaches often fall short in adequately identifying and mitigating these vulnerabilities, leaving websites susceptible to exploitation by malicious actors.

Furthermore, many website owners lack the necessary tools and expertise to conduct comprehensive security assessments, resulting in gaps in their defense strategies. As a result, there is a pressing need for an advanced website vulnerability analysis tool that can systematically identify and address potential security risks, empowering website owners to fortify their defenses and safeguard their digital assets.

WebGuard aims to address this challenge by providing a robust and user-friendly platform for conducting thorough website security assessments. By leveraging advanced scanning techniques and providing actionable insights, WebGuard enables users to proactively identify vulnerabilities and implement effective security measures, thereby enhancing the overall resilience of their web infrastructure.

1.2. THE SOLUTION

The solution offered by WebGuard is a comprehensive website vulnerability analysis tool designed to empower website owners and administrators to enhance their cybersecurity posture. By amalgamating advanced scanning techniques with intuitive user interface design, WebGuard provides users with the means to conduct thorough security assessments of their web infrastructure.

Through WebGuard's functionality, users can input target URLs or IP addresses and specify additional ports to scan, allowing for tailored and comprehensive security evaluations. The tool meticulously examines websites for common vulnerabilities such as open ports, outdated software, weak authentication mechanisms, and missing security headers.

Moreover, WebGuard doesn't just identify vulnerabilities; it also provides actionable recommendations to mitigate risks and improve overall security posture. This proactive approach equips users with the insights and tools needed to fortify their web defenses against evolving cyber threats effectively.

In summary, WebGuard offers a robust solution for website vulnerability analysis, empowering users to identify and address security weaknesses proactively, thereby bolstering the resilience of their web infrastructure and safeguarding their digital assets.

1.3. EXISTING SYSTEM

1.3.1. EXISTING CONCEPT

Existing website vulnerability analysis tools often lack the comprehensiveness and user-friendliness required to effectively address the evolving landscape of cybersecurity threats. While some tools focus on specific vulnerabilities or aspects of website security, they may not provide a holistic assessment of potential risks. Additionally, many existing tools require significant technical expertise to operate, limiting their accessibility to website owners and administrators without specialized knowledge. Moreover, the lack of actionable recommendations and intuitive interfaces further hinders the effectiveness of these tools. Overall, there is a need for an advanced website vulnerability analysis tool that combines comprehensive scanning capabilities with user-friendly features to empower website owners to proactively identify and mitigate security risks.

1.3.2. DRAWBACKS

- Technical expertise required.
- Lack of comprehensive assessment.
- Limited actionable recommendations.
- Inaccessible to non-experts.

1.4. PROPOSED SYSTEM

1.4.1. PROPOSED CONCEPT

The proposed concept aims to address the limitations of existing website vulnerability analysis tools by offering a user-friendly and comprehensive solution accessible to users of all technical backgrounds. The concept incorporates advanced scanning techniques to identify a wide range of vulnerabilities, including open ports, outdated software, weak authentication mechanisms, and missing security headers. Additionally, the tool provides actionable recommendations tailored to each identified vulnerability, enabling users to implement effective security measures proactively. The intuitive interface simplifies the scanning process, allowing users to input target URLs or IP addresses and specify additional ports with ease. By combining accessibility, comprehensiveness, and usability, the proposed concept seeks to empower website owners and administrators to fortify their web infrastructure against cyber threats effectively.

1.4.2. ADVANTAGES

- User-friendly interface.
- Comprehensive vulnerability assessment.
- Actionable security recommendations.
- Accessibility to non-experts.

LITERATURE SURVEY

CHAPTER 2

2. LITERATURE SURVEY

Open Source Solutions for Vulnerability Assessment: A Comparative Analysis

Authors: Dinis Barroqueiro Cruz

Year: 2023

As software applications continue to become more complex and attractive to cyber-attackers, enhancing resilience against cyber threats becomes essential. Aiming to provide more robust solutions, different approaches were proposed for vulnerability detection in different stages of the application lifecycle. This article explores three main approaches to application security: Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and Software Composition Analysis (SCA). The analysis conducted in this work is focused on open-source solutions while considering commercial solutions to show contrast in the approaches taken and to better illustrate the different options available. It proposes a baseline comparison model to help evaluate and select the best solutions, using comparison criteria that are based on community standards. This work also identifies future opportunities for application security, highlighting some of the key challenges that still need to be addressed in order to fully protect against emerging threats, and proposes a workflow that combines the identified tools to be used for vulnerability assessments.

2.1. SOME OTHER RESEARCHERS

Title : VulScan: A Web-Based Vulnerability Multi-Scanner for Web Application

Authors: Tobias Osemegbe Odion; Ife Olalekan Ebo; Rajab Mohammed Imam; Abdullahi Isa Ahmed; Usman Nuhu Musa

Year: 2023

Web application is becoming more powerful and widely used as more people are accessing it. However, web application attacks have become an ongoing problem as a result of one or more vulnerabilities that may be present. Most researchers have been able to propose solutions capable of handling specific attacks, which are not capable of identifying the vulnerabilities present in today's dynamic web applications. Hence, the proposed system named “Vulnerability Scanner (VulScan)” has the ability to identify six distinct categories of attacks that web applications may be susceptible to. The system uses an authenticated scanning approach to scan web application components such as links, forms, headers etc., and reports susceptible vulnerabilities. The detection capability of VulScan was evaluated on Damn Vulnerable Web Application (DVWA) and live web applications.

Title : An Integrated Vulnerability Assessment Tool for Web Applications

Authors: Stewart Kirubakaran S; G. Jaspher W Kathrine; Arul Xavier V M; G. Mathew Palmer

Year: 2022

Security criteria fixation and analysis for all the web applications developed and deployed on the internet is very much required due to the excess usage of many users accessing the application. The application has to respond in a fraction of nanoseconds for users to remain attached to the application. In this scenario, the need to ensure the basic security requirements of Confidentiality, Integrity, and Availability is an important task. To analyze the security requirements penetration testing is being done. The process of manual penetration testing is slowly shifting towards automated testing systems. Security experts are designing automation tools to detect vulnerabilities. This paper deals with binding the known multiple vulnerabilities to find the overall vulnerability of a web application. The impact of the vulnerability is analyzed by combining multiple penetration testing tools which automatically classify the exploit model to gather information using the signature patterns (Regular Expressions). Finally, the exploit information can be viewed in the form of a report so that security patches of the vulnerabilities can be performed.

Title : An Automatic Vulnerability Scanner for Web Applications

Author: Haibo Chen; Junzuo Chen; Jinfu Chen; Shang Yin; Yiming Wu; Jiaping

Year: 2020

With the progressive development of web applications and the urgent requirement of web security, vulnerability scanner has been particularly emphasized, which is regarded as a fundamental component for web security assurance. Various scanners are developed with the intention of that discovering the possible vulnerabilities in advance to avoid malicious attacks. However, most of them only focus on the vulnerability detection with single target, which fail in satisfying the efficiency demand of users. In this paper, an effective web vulnerability scanner that integrates the information collection with the vulnerability detection is proposed to verify whether the target web application is vulnerable or not. The experimental results show that, by guiding the detection process with the useful collected information, our tool achieves great web vulnerability detection capability with a large scanning scope.

Title : Analysis of Web Security Using Open Web Application Security Project

Author: Muhamad Agreindra Helmiawan; Esa Firmansyah; Irfan Fadil; Yanvan Sofivan; Fathoni Mahardika; Agun Guntara

Year: 2020

Open Web Application Security Project 10 is a web application security testing framework method that focuses on web application security to find weaknesses in a website. The Open Web Application Security Project 10 aims to ensure the safety of websites in form checklists. Open Web Application Security Project 10 has the ten most dangerous types of website vulnerabilities such as injection, broken authentication, sensitive data exposure, Extensible Markup Language external entities, corrupted access control, security misconfiguration, cross-site scripting, unreliable deserialization, segment exploitation with known weaknesses, and lack of logging and checking. This paper analyzes and tests the security of the web along with six sub-domains with the aim of knowing and assessing the security level of a website, whether additional security is needed, and recommendations on the website. The results of this paper indicate that the web has a security level 80%, web informatics engineering subdomain 60%, information systems 60%, informatics management 60%, integrated academic system 80%, student acceptance 80% and e-learning 80%.

SYSTEM SPECIFICATION

CHAPTER 3

3. SYSTEM SPECIFICATION

3.1. HARDWARE REQUIREMENT

- **SYSTEM** : Intel i5 processor
- **HARDDISK** : 50 GB
- **MONITOR** : 15'' LED
- **INPUT DEVICES** : Keyboard, Mouse.
- **RAM** : 4 GB
- **CONNECTIVITY** : LAN or Wi-Fi

3.2. SOFTWARE REQUREMENT

- **FRONT END** : Python GUI
- **SOFTWARE USED** : VS Code editor
- **LANGUAGE** : Python
- **MODULES** : tkinter,nmap,json,socket,beatifulsoup.
- **EDITING TOOLS** : Notepad++,Python IDLE (3.0)

PROJECT IMPLEMENTATION

CHAPTER 4

4. PROJECT IMPLEMENTATION

4.1. MODULES

- User Interface
- Scanning Engine
- Vulnerability Analysis
- Security Recommendations
- Concurrency Management

4.1.1. USER INTERFACE

This part encompasses the graphical user interface (GUI) developed using the **tkinter** module. It provides the interface for users to interact with the application, input target URLs or IP addresses, specify additional ports, initiate scans, and view scan results.

4.1.2. SCANNING ENGINE

This part involves the core scanning functionality of the application. It utilizes the nmap module to perform network scans, identify open ports, retrieve information about services running on those ports, and assess vulnerabilities based on the scan results.

4.1.3. VULNERABILITY ANALYSIS

This part includes the analysis of scan results to identify potential vulnerabilities. It checks for common security issues such as open ports, outdated software, weak authentication mechanisms, and missing security headers.

4.1.4. SECURITY RECOMMENDATIONS

Based on the vulnerabilities detected, this part provides actionable recommendations to mitigate risks and enhance security posture. It offers suggestions such as closing unnecessary ports, updating software, enforcing strong password policies, and implementing security headers.

4.1.5. CONCURRENCY MANAGEMENT

This part involves managing concurrency to ensure smooth operation of the application. It utilizes the threading module to handle concurrent tasks, particularly during the scanning process, to prevent freezing of the UI and maintain responsiveness.

SYSTEM DESIGN

CHAPTER 5

5. SYSTEM DESIGN

5.1. GENERAL

The WebGuard project integrates an automated scanning subsystem leveraging the `nmap` module for thorough network scans and vulnerability identification. Its vulnerability analysis subsystem assesses scan results to pinpoint security risks such as open ports, outdated software, and weak authentication mechanisms. Recommendations for risk mitigation are provided based on detected vulnerabilities. The user-friendly interface, developed with `tkinter`, enables seamless interaction, empowering users to input target URLs, initiate scans, and access comprehensive reports effortlessly. Concurrency management ensures smooth operation during scans, enhancing responsiveness. WebGuard offers a scalable, efficient, and user-centric solution for bolstering website security in diverse environments.

Inputs to System Design

System design takes the following inputs -

1. **Target URL or IP Address:**

   Users input the URL or IP address of the website they want to scan for vulnerabilities. This serves as the primary input for initiating the scanning process.

2. **Additional Ports (Optional):**

   Users have the option to specify additional ports they want to scan apart from the default ports. This input allows for customization of the scanning process based on specific requirements.

Outputs for System Design

System design gives the following outputs -

1. **Scan Results:**

   The system provides a detailed report of the scan results, including information about open ports, services running on those ports, and any identified vulnerabilities**.**

2. **Vulnerability Analysis Report:**

   Users receive a comprehensive analysis of the vulnerabilities detected during the scan, highlighting potential security risks such as open ports, outdated software, weak authentication mechanisms, and missing security headers.

3. **Security Recommendations:**

   Based on the vulnerabilities identified, the system generates actionable recommendations for mitigating risks and enhancing the security posture of the target website. These recommendations may include closing unnecessary ports, updating software, enforcing strong password policies, and implementing security headers.

4. **Real-time Feedback (Optional):**

   If configured, the system can provide real-time feedback to users during the scanning process, highlighting progress, detected vulnerabilities, and any immediate actions that need to be taken.

5.2. SYSTEM ARCHITECTURE

A system architecture is a conceptual model that outlines a system's structure, behaviour, and additional viewpoints. An architectural description is a formal description and representation of a system that is arranged in a way that allows for reasoning about the system's structures and actions.

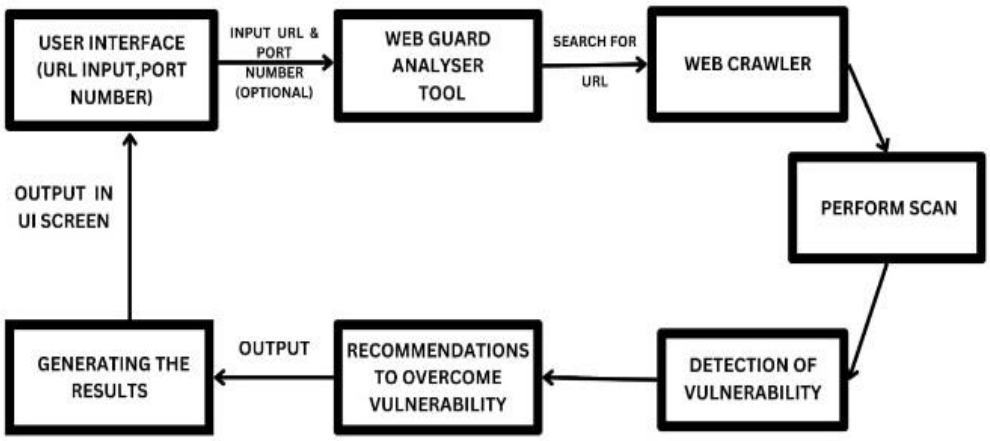


Fig 5.1. SYSTEM ARCHITECTURE

5.3. USECASE DIAGRAM

A use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams

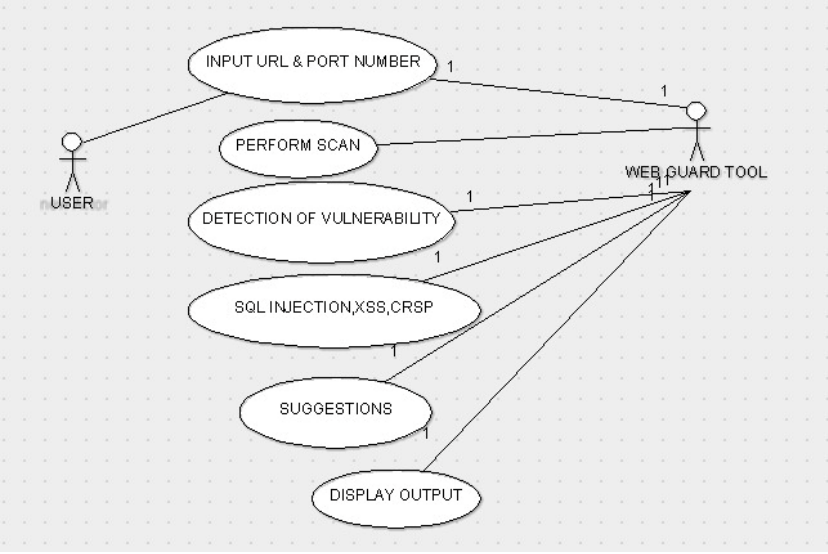


Fig 5.2. USECASE DIAGRAM

5.4 CLASS DIAGRAM

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object- oriented languages and thus widely used at the time of construction.

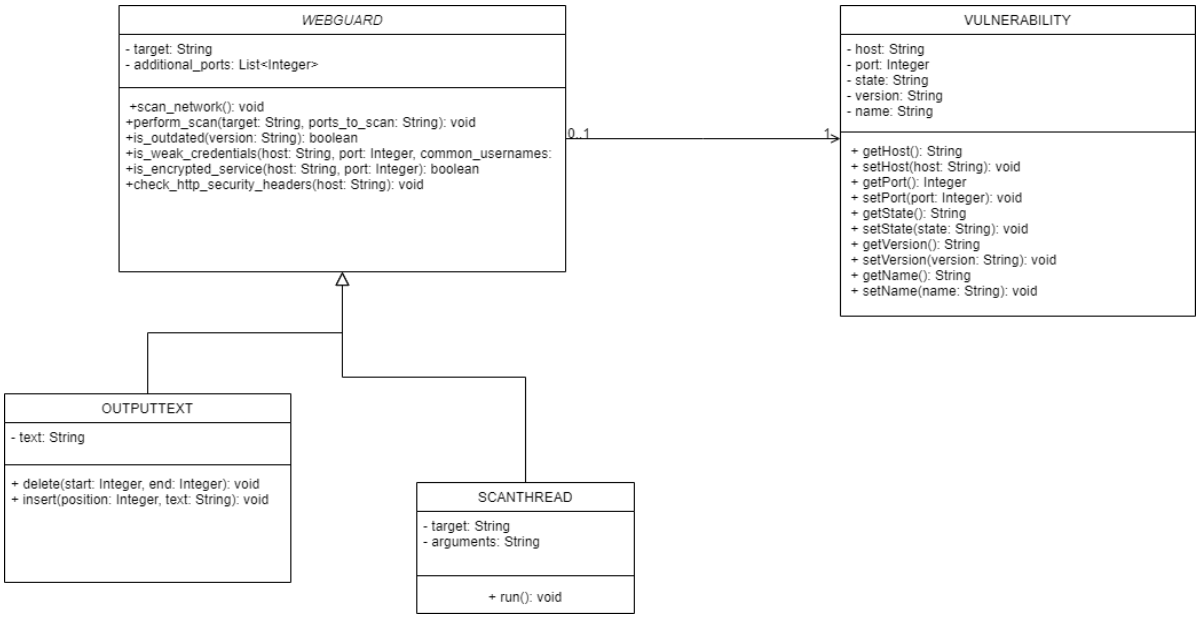


Fig 5.3. CLASS DIAGRAM

5.5 ACTIVITY DIAGRAM

An activity diagram is essentially a flowchart that shows how one activity leads to another. The action can be referred to as a system operation. One operation leads to the next in the control flow. This flow may be parallel, concurrent, or branched.

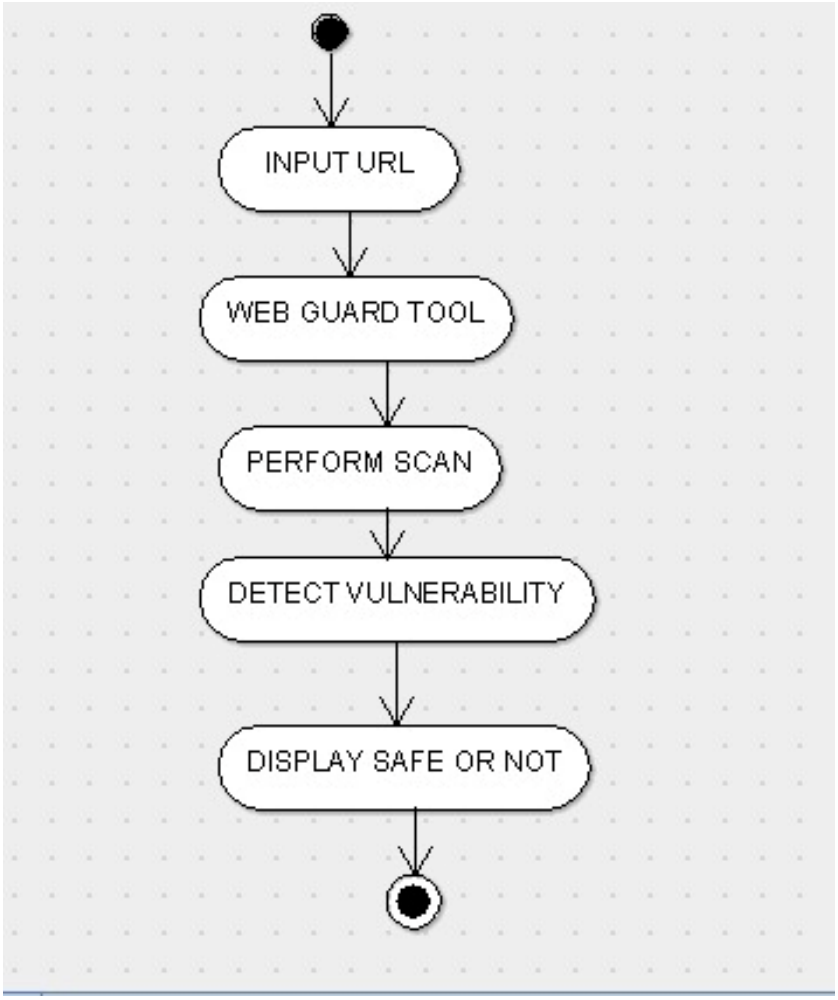


Fig 5.4. ACTIVITY DIAGRAM

5.6 STATE DIAGRAM

A state diagram, also known as a state machine diagram, depicts the various states that an object or system can be in, as well as the transitions between these states. Each state represents a condition or situation in the system, and transitions occur in response to events or stimuli. State diagrams are composed of states, transitions, initial and final states, and can effectively model the behavior and lifecycle of systems or entities.

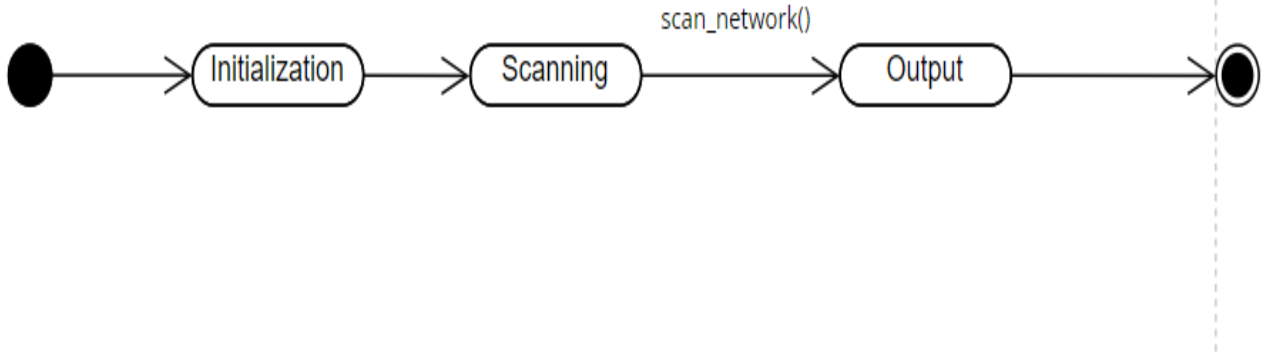


Fig 5.5.STATE DIAGRAM

SOFTWARE SPECIFICATION

CHAPTER 6

6. SOFTWARE SPECIFICATION

6.1. GENERAL

Software Specification is a list of Software, Packages, libraries required to develop a project. Software Specification contains the deep description about the project.

6.2. PYTHON

6.2.1. INTRODUCTION TO PYTHON

Python is a highly popular high-level programming language renowned for its readability, simplicity, and versatility. Its syntax is designed with code readability in mind, using whitespace and indentation to structure the code, resulting in clean and easily understandable code. Python appeals to both experienced programmers seeking clarity and maintainability and beginners learning programming fundamentals.

One of Python's notable advantages is its extensive standard library, offering a wide range of modules and functions. This eliminates the need for writing additional code, making it efficient and convenient for performing various tasks.

Python's versatility is demonstrated by its applicability across diverse domains, including web development, scientific computing, data analysis, artificial intelligence, and machine learning. It boasts a rich ecosystem of libraries and frameworks, enabling developers to extend its functionality and integrate it seamlessly with other tools.

6.2.2.FEATURES OF PYTHON

Python is a versatile and powerful programming language with a variety of features that make it popular among developers. Here are some of the key features of Python:

- **Easy to Learn**: Python is a beginner-friendly language that is easy to learn and understand. Its simple syntax and emphasis on readability make it an ideal language for new programmers.
- **Interpreted**: Python is an interpreted language, which means that code is executed line by line at runtime. This makes it easier to debug and test code, as changes can be made and tested immediately without the need to recompile the entire program.
- **Object-Oriented**: Python is an object-oriented language, which means that it supports the creation of classes and objects, encapsulation, inheritance, and polymorphism.
- **High-level Language**: Python is a high-level language, which means that it abstracts away many low-level details, such as memory management and system calls. This makes it easier to write code and focus on solving problems rather than worrying about low-level details.
- **Cross-platform**: Python is a cross-platform language, which means that it can run on different operating systems, such as Windows, Linux, and macOS.
- **Large Standard Library**: Python has a vast standard library that contains many useful modules and functions for performing a wide range of tasks, such as file I/O, networking, and regular expressions.
- **Third-Party Libraries:** Python has a rich ecosystem of third-party libraries and frameworks that extend the language's functionality, such as NumPy, Pandas, Django, Flask, and TensorFlow.

6.2.3. OBJECTIVES OF PYTHON

Python is a robust and functional programming language that serves a wide range of functions. Its main goals are to be productive, adaptable, and simple to learn. Here are a few sentences that go into greater detail about Python's goals:

Python's main goal is to be simple to use and learn. It's a great option for beginning programmers thanks to its clear, simple syntax that prioritizes readability. Python's design ethos, which places a high value on simplicity and minimalism, makes the language simple to learn and retain. Additionally, Python has a sizable standard library that is filled with numerous built-in functions and modules that facilitate common programming tasks.

Being productive and flexible is another goal of Python. The Python design philosophy emphasizes the value of modularity, extensibility, and reuse of code. Because of this, writing code that is reusable and simple to maintain is simple. Python's dynamic typing system eliminates the need for type declarations, which speeds up code writing compared to languages with static typing. Additionally, Python has a large community of third-party libraries and frameworks that make it simple to expand the language's functionality for particular tasks like data analysis, web development, and machine learning. Python is a highly productive language that is suitable for a variety of programming tasks thanks to all of these features.

6.2.4. PYTHON FRAMEWORKS

Python is a strong and adaptable programming language that is used in a variety of applications, including machine learning, data analysis, scientific computing, and web development. Developers have developed a variety of frameworks that offer pre-built code and tools for specific programming tasks to make programming with Python simpler and more effective. Here are a few sentences discussing Python frameworks:

Python frameworks are collections of pre-written tools and code that make it simpler to create sophisticated applications. By providing pre-built code for common tasks, they can save time and effort for developers by giving them a structure and set of guidelines for creating applications. Django, Flask, and Pyramid are just a few of the well-liked web development frameworks available in Python. These frameworks offer a variety of tools and features, including database management, routing, and templating, which make it simpler to create dependable and scalable web applications.

Python frameworks are also frequently used in scientific computing and data analysis. Numerous tools and functions are available for working with large datasets, conducting statistical analysis, and visualizing data in Python frameworks like NumPy, SciPy, and Pandas. These frameworks simplify complex calculations and analysis for researchers and scientists, and they can be applied to a variety of disciplines, including biology, engineering, physics, and finance. Artificial intelligence and machine learning can both benefit from the use of Python frameworks. Machine learning models can be built and trained using frameworks like TensorFlow and PyTorch, which are popular in both academia and business.

6.2.5. PACKAGES OF PYTHON

Some of the python packages used in this project are,

1. **tkinter:** Used for creating the graphical user interface (GUI) of the application.

2. **nmap:** Utilized for network scanning to detect open ports and retrieve information about services running on those ports.

3. **json:** Used for handling JSON data, particularly for loading common passwords from a JSON file.

4. **requests:** Employed for making HTTP requests to retrieve data from external sources, such as retrieving common usernames from a URL.

5. **socket:** Utilized for creating network sockets, particularly in the is_encrypted_service function for checking if a service is encrypted.

6. **paramiko:** Used for SSH protocol implementation, particularly in the is_weak_credentials function for checking weak SSH credentials.

7. **threading:** Utilized for managing threads, particularly in handling the scanning process concurrently with the GUI operations.

SOFTWARE TESTING

CHAPTER 7

7.SOFTWARE TESTING

7.1.GENERAL

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing. is to identify errors, gaps or missing requirements in contrast to actual requirements.

Some prefer saying Software testing definition as a White Box and Black Box Testing. In simple terms, Software Testing means the Verification of Application Under Test. This Software Testing course introduces testing software to the audience and justifies the importance of software testing.

Importance of Software Testing

Software Testing is Important because if there are any bugs or errors in the software, it can be identified early and can be solved before delivery of the software product. Properly tested software product ensures reliability, security and high performance which further results in time saving, cost effectiveness and customer satisfaction.

Need of Testing

Testing is important because software bugs could be expensive or even dangerous. Software bugs can potentially cause monetary and human loss, and history is full of such examples.

Here are the benefits of using software testing:

1. **Cost-Effective**: It is one of the important advantages of software testing. Testing any IT project on time helps you to save your money for the long term. In case if the bugs caught in the earlier stage of software testing, it costs less to fix.
2. **Security**: It is the most vulnerable and sensitive benefit of software testing. People are looking for trusted products. It helps in removing risks and problems earlier.

7.2.TYPES OF TESTING

7.2.1. UNIT TESTING

A unit test is a way of testing a unit - the smallest piece of code that can be logically isolated in a system. In most programming languages, that is a function, a subroutine, a method or property. The isolated part of the definition is important. In his book "Working Effectively with Legacy Code", author Michael Feathers states that such tests are not unit tests when they rely on external systems: "If it talks to the database, it talks across the network, it touches the file system, it requires system configuration, or it can't be run at the same time as any other test."

Modern versions of unit testing can be found in frameworks like JUnit, or testing tools like Test Complete. Look a little further and you will find a Unit, the mother of all unit testing frameworks created by Kent Beck, and a reference in chapter 5 of The Art of Software Testing. Before that, it's mostly a mystery. I asked Jerry Weinberg about his experiences with unit testing -- "We did unit testing in 1956. As far as I knew, it was always done, as long as there were computers".

7.2.2. FUNCTIONAL TESTING

Functional testing is a sort of black-box testing that relies its test cases on the requirements of the software component being tested. Internal program structure is rarely considered when testing functions by feeding them input and reviewing the result. Functional Testing is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.

This testing checks User Interface, APIs, Database, Security, Client/Server communication and other functionality of the Application Under Test. The testing can be done either manually or using automation.

7.2.3. SYSTEM TESTING

System testing is a sort of software testing that is conducted on a whole integrated system to assess the system's compliance with the associated requirements. Passed integration testing components are used as input in system testing. The purpose of integration testing is to discover any discrepancies between the components that are being integrated together. System testing finds flaws in both integrated modules and the entire system. The observed behavior of a component or system when tested is the result of system testing. System testing is performed on the entire system in the context of either system requirement specifications, functional requirement specifications, or both. System testing examines the system's design and behavior, as well as the

customer's expectations. It is carried out to test the system beyond the parameters specified in the software requirements specification.

System testing is mostly carried out by a testing team that is independent of the development team and helps to assess the system's quality impartially. It is subjected to both functional and non-functional testing. Black-box testing is used in system testing. System testing comes after integration testing but before acceptance testing.

7.2.4. PERFORMANCE TESTING

Performance testing is a type of testing that assesses the speed, responsiveness, and stability of a computer, network, software program, or device when subjected to a workload. Organizations will conduct performance testing to discover performance bottlenecks.

Without some form of performance testing in place, system performance will likely be affected with slow response times, experiences that are inconsistent between users and the operating system, creating an overall poor user experience. Determining if the developed system meets speed, responsiveness and stability requirements while under workloads will help ensure a more positive user experience.

Performance testing can involve quantitative tests done in a lab, or in some scenarios, occur in the production environment. Performance requirements should be identified and tested. Typical parameters include processing speed, data transfer rates, network bandwidth and throughput, workload efficiency and reliability. As an example, an organization can measure the response time of a program when a user requests an action; the same can be done at scale.

7.2.5. INTEGRATION TESTING

Integration testing comes after unit testing in which, units or individual components of the software are rested in a group. The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units.

The Software is developed with a number of software modules that are coded by different coders or programs. The goal of integration testing is to check the correctness of communication among all the modules.

7.2.6. ACCEPTANCE TESTING

Acceptance testing is formal testing it determines whether the software is conforming specified requirements and user requirements or not. It is conducted as a kind of Black Box testing where the number of required users involved testing the acceptance level of the system. It is the fourth and last level of software testing.

User acceptance testing is a type of testing, which is done by the customer before accepting the final product. Generally, it is done by the customer (domain expert) for their satisfaction, and check whether the application is working according to given business scenarios, real-time scenarios.

In this, we concentrate only on those features and scenarios which are regularly used by the end-user or the customer. However, the software has passed through three testing levels (Unit Testing, Integration Testing, System Testing) But still there are some minor errors which can be identified when the system is used by the end user in the actual scenario.

CODING

CHAPTER 8

8.CODING

8.1.WEB-VULNERABILITY.PY

```
import tkinter as tk
import beatifulsoup
from tkinter import ttk, messagebox
import nmap
import json
import requests
import socket
import paramiko
import threading
passwords_url = 'https://raw.githubusercontent.com/vlhomme/list-of-most-common-password/master/passwords.json'
passwords_response = requests.get(passwords_url)
common_passwords = json.loads(passwords_response.text)
usernames_url = 'https://gist.githubusercontent.com/kivox/920c271ef8dec2b33c84e1f2cc2977fc/raw/77e5d15c626f3da529e4c78aa3c1b3203b8d8dbb/common_usernames.txt'
usernames_response = requests.get(usernames_url)
common_usernames = usernames_response.text.splitlines()
def scan_network():
    target = target_entry.get()
    if not target:
```

```
        messagebox.showerror("Error", "Please enter a target (URL/IP) to scan.")
        return
    additional_ports = additional_ports_entry.get()
    ports_to_scan = ['1-1000', '1433']  # Default ports to scan
    if additional_ports:
        additional_ports_list = additional_ports.split(',')
        valid_ports = []
        for port_spec in additional_ports_list:
            if '-' in port_spec:
                # Check if the port specification is a range
                start, end = port_spec.split('-')
                try:
                    start_port = int(start)
                    end_port = int(end)
                    if start_port <= end_port and 1 <= start_port <= 65535 and 1 <= end_port
<= 65535:
                        valid_ports.append(port_spec)
                    else:
                        raise ValueError("Invalid port range: start and end ports must be
between 1 and 65535.")
                except ValueError:
                    pass  # Ignore if the port range cannot be converted to integers
            else:
                # Check if the port specification is a single port number
                try:
                    port_number = int(port_spec)
```

```
                if 1 <= port_number <= 65535:  # Valid port range
                    valid_ports.append(port_spec)
                else:
                    raise ValueError("Invalid port number: port must be between 1 and 65535.")
            except ValueError:
                pass  # Ignore if the port number cannot be converted to an integer
        if valid_ports:
            ports_to_scan.extend(valid_ports)
            arguments = '-p ' + ','.join(ports_to_scan)
        else:
            messagebox.showerror("Error", "No valid ports specified. Please enter valid port numbers or ranges.")
            return
    else:
        arguments = '-p ' + ','.join(ports_to_scan)
    output_text.delete(1.0, tk.END)
    output_text.insert(tk.END, "Security check initiated....\n","info")
    for thread in threading.enumerate():
        if thread.name == "ScanThread":
            messagebox.showwarning("Warning", "A scan is already in progress. Please wait for it to complete.")
            return
    scan_thread = threading.Thread(name="ScanThread", target=perform_scan, args=(target, arguments))
    scan_thread.start()
```

```
def perform_scan(target, ports_to_scan):
    # Join the elements of ports_to_scan list into a single string
    arguments = ' '.join(ports_to_scan)
    scanner = nmap.PortScanner()
    scanner.scan(target, arguments=arguments)
    output_text.delete(1.0, tk.END)
    output_text.insert(tk.END, "Security check initiated....\n","info")
    safe_website = True  # Assume the website is safe by default
    for host in scanner.all_hosts():
        output_text.insert(tk.END, f"Scanning host: {host}\n", "info")
        for proto in scanner[host].all_protocols():
            output_text.insert(tk.END, f"  Protocol: {proto}\n", "info")
            ports = scanner[host][proto].keys()
            for port in ports:
                port_info = scanner[host][proto][port]
                output_text.insert(tk.END, f"    Port: {port} - State: {port_info['state']}\n",
"info")
                if port_info['state'] == 'open':
                    output_text.insert(tk.END, "      Potential vulnerability found: Open
port\n","warning")
                    output_text.insert(tk.END, "      Suggestion: Close unnecessary ports to
reduce attack surface\n","warning")
                    safe_website = False
                if 'version' in port_info and port_info['version'].lower() != 'unknown':
                    version = port_info['version']
```

```
                output_text.insert(tk.END, f"      Software version: {version}\n","info")
                if is_outdated(version):
                    output_text.insert(tk.END, "      Potential vulnerability found: Outdated
software\n","warning")
                    output_text.insert(tk.END, "      Suggestion: Update software to the
latest version\n","warning")
                    safe_website = False
            if 'ssh' in port_info['name']:
                if is_weak_credentials(host, port, common_usernames,
common_passwords):
                    output_text.insert(tk.END, "      Potential vulnerability found: Weak
SSH credentials\n","warning")
                    output_text.insert(tk.END, "      Suggestion: Enforce strong password
policies\n","warning")
                    safe_website = False
            if port in [80, 443]:
                if not is_encrypted_service(host, port):
                    output_text.insert(tk.END, f"      Potential vulnerability found:
Unencrypted service on port {port}\n","warning")
                    output_text.insert(tk.END, "      Suggestion: Enable encryption (e.g.,
SSL/TLS) for sensitive services\n","warning")
                    safe_website = False
                    if port == 80:
                        check_http_security_headers(host)
            elif port == 21:
                if not is_encrypted_service(host, port):
```

```
                output_text.insert(tk.END, f"      Potential vulnerability found:
Unencrypted service on port {port}\n","warning")
                output_text.insert(tk.END, "      Suggestion: Avoid using FTP and
switch to SFTP or FTPS for secure file transfer\n","warning")
                safe_website = False
            elif port == 23:
                output_text.insert(tk.END, f"      Potential vulnerability found:
Unencrypted service on port {port}\n","warning")
                output_text.insert(tk.END, "      Suggestion: Avoid using Telnet due to
security risks. Switch to SSH for secure remote access\n","warning")
                safe_website = False
            elif port == 1433:
                output_text.insert(tk.END, f"      Potential vulnerability found: MSSQL
detected on port {port}\n","warning")
                output_text.insert(tk.END, "      Suggestion: Ensure MSSQL is properly
configured to prevent SQL injection attacks\n","warning")
                safe_website = False
    if safe_website:
        output_text.insert(tk.END, "No significant vulnerabilities found. Positive aspects
about the website:\n", "safe")
        output_text.insert(tk.END, "- Secure communication via HTTPS\n", "safe")
        output_text.insert(tk.END, "- Minimal attack surface\n", "safe")
        output_text.insert(tk.END, "- Up-to-date software\n", "safe")
        output_text.insert(tk.END, "- No weak SSH credentials found.\n", "safe")
        output_text.insert(tk.END, "- No unencrypted services found.\n", "safe")
        output_text.insert(tk.END, "- No SQL injection vulnerabilities found.\n", "safe")
```

```
        output_text.insert(tk.END, "- Strong authentication\n", "safe")
        output_text.insert(tk.END, "- No missing security headers found.\n", "safe")
        output_text.insert(tk.END, "- Adherence to security best practices\n", "safe")
        output_text.insert(tk.END, "- User trust and credibility\n", "safe")
        output_text.insert(tk.END, "No significant vulnerabilities found. The website
appears to be safe.\n", "safe")
    else:
        output_text.insert(tk.END, "Vulnerabilities detected! Review the suggestions
provided to improve security.\n", "warning")
    output_text.insert(tk.END, "Scan complete.\n")
def is_outdated(version):
    latest_version = '1.0.0'
    return version < latest_version
def is_weak_credentials(host, port, common_usernames, common_passwords):
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    for username in common_usernames:
        for password in common_passwords:
            try:
                ssh.connect(host, port=port, username=username, password=password,
timeout=5)
                return True
            except paramiko.AuthenticationException:
                pass
            except Exception as e:
                print(f"Error: {e}")
```

```
    ssh.close()
    return False

def is_encrypted_service(host, port):
    if port == 80:
        try:
            with socket.create_connection((host, port)) as sock:
                sock.send(b'GET / HTTP/1.1\r\nHost: example.com\r\n\r\n')
                response = sock.recv(1024).decode('utf-8')
                if 'Location: https://' in response:
                    return True
                else:
                    return False
        except Exception as e:
            print(f"Error checking port {port}: {e}")
            return False
    elif port == 443:
        return True
    else:
        return True
def check_http_security_headers(host):
    try:
        response = requests.get(f"http://{host}")
        headers = response.headers
        if 'X-Frame-Options' not in headers:
            output_text.insert(tk.END, "      Potential vulnerability found: Missing X-
```

```
Frame-Options header\n", "warning")
            output_text.insert(tk.END, "      Suggestion: Implement X-Frame-Options
header to prevent Clickjacking attacks\n", "warning")
        if 'X-XSS-Protection' not in headers:
            output_text.insert(tk.END, "      Potential vulnerability found: Missing X-XSS-
Protection header\n", "warning")
            output_text.insert(tk.END, "      Suggestion: Implement X-XSS-Protection
header to mitigate XSS attacks\n", "warning")
        if 'Strict-Transport-Security' not in headers:
            output_text.insert(tk.END, "      Potential vulnerability found: Missing Strict-
Transport-Security header\n", "warning")
            output_text.insert(tk.END, "      Suggestion: Implement Strict-Transport-
Security header to enforce HTTPS\n", "warning")
        if 'Content-Security-Policy' not in headers:
            output_text.insert(tk.END, "      Potential vulnerability found: Missing
Content-Security-Policy header\n", "warning")
            output_text.insert(tk.END, "      Suggestion: Implement Content-Security-
Policy header to prevent various types of attacks\n", "warning")
    except Exception as e:
        print(f"Error checking HTTP headers: {e}")
root = tk.Tk()
root.title("WEB GUARD-Advance Website Vulnerability Scanner")
target_label = ttk.Label(root, text="Enter Target (URL/IP):")
target_label.pack(pady=5)
target_entry = ttk.Entry(root, width=75)
target_entry.pack()
```

```
additional_ports_label = ttk.Label(root, text="Additional Ports to Scan (comma-separated):"+"0 - 65535")
additional_ports_label.pack(pady=5)
additional_ports_entry = ttk.Entry(root, width=75)
additional_ports_entry.pack()
scan_button = ttk.Button(root, text="Scan", command=scan_network)
scan_button.pack(pady=10)
output_text = tk.Text(root, width=80, height=20)
output_text.pack(fill=tk.BOTH, expand=True)
output_text.tag_config("safe", foreground="green")
output_text.tag_config("warning", foreground="red")
output_text.tag_config("info", foreground="black")
root.mainloop()
```

SCREENSHOT

CHAPTER 9

9.SCREENSHOT

9.1.GRAPHICAL USER INTERFACE WINDOW

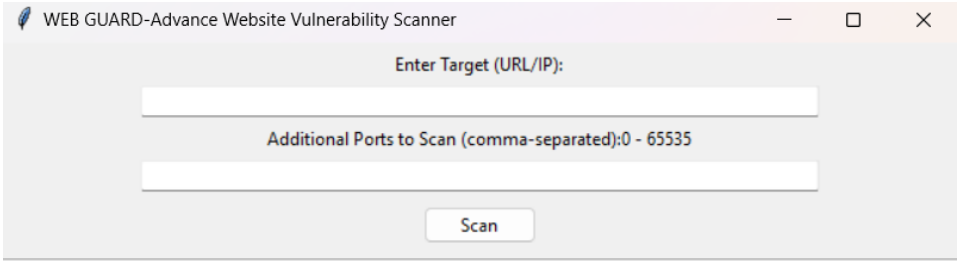


Fig 9.1. GRAPHICAL USER INTERFACE WINDOW

9.2.ALERT BOX WHEN NO URL IS ENTERED

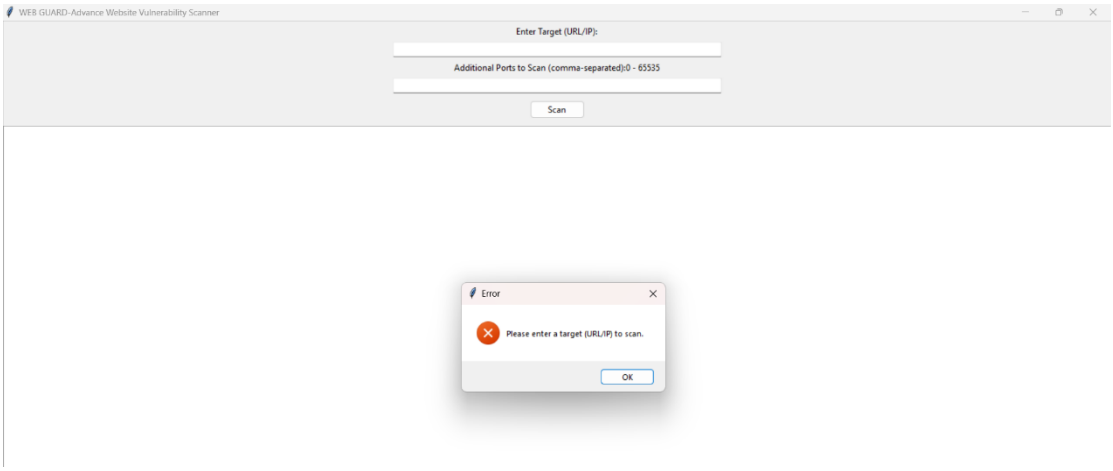
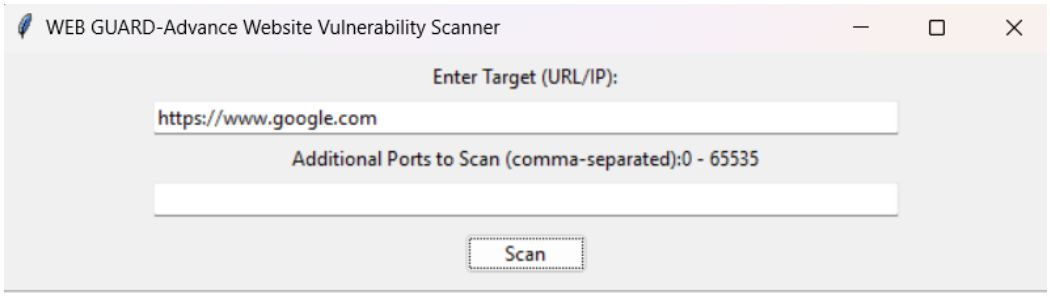


Fig 9.2. Alert box when no URL is entered

9.3.OUTPUT




```
Security check initiated....
No significant vulnerabilities found. Positive aspects about the website:

- Secure communication via HTTPS
- Minimal attack surface
- Up-to-date software
- No weak SSH credentials found.
- No unencrypted services found.
- No SQL injection vulnerabilities found.
- Strong authentication
- No missing security headers found.
- Adherence to security best practices
- User trust and credibility

No significant vulnerabilities found. The website appears to be safe.
Scan complete.
```

Fig 9.3. OUTPUT-1 (SECURE WEBSITE)

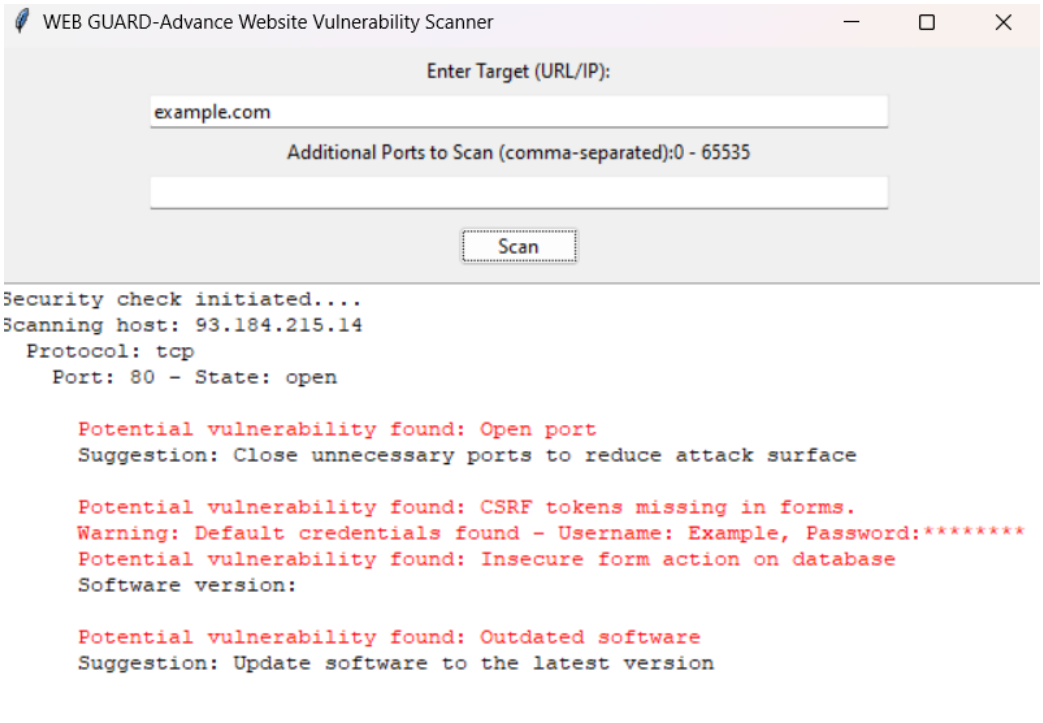


Fig 9.4. OUTPUT -2 (INSECURE WEBSITE)

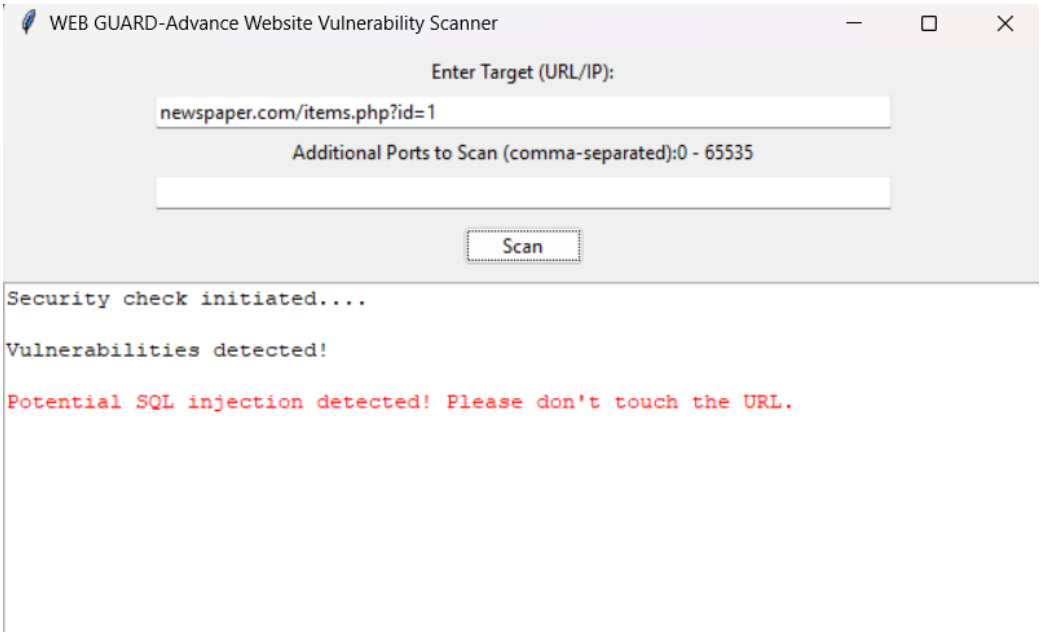


Fig 9.5. OUTPUT -3 (SQL INJECTED WEBSITE)

CONCLUSION

CHAPTER 10

10.CONCLUSION

In conclusion, the WebGuard project represents a significant advancement in the realm of website vulnerability analysis, providing a comprehensive and user-friendly solution for assessing and enhancing the security of web infrastructure. By integrating automated scanning capabilities, vulnerability analysis algorithms, and actionable security recommendations, WebGuard empowers users to proactively identify and mitigate potential risks, ultimately bolstering the resilience of their websites against cyber threats. The system's scalability, efficiency, and accuracy make it suitable for various educational, corporate, and organizational environments, ensuring consistent and reliable evaluations across different contexts. By automating the traditionally labor-intensive task of website security assessment, WebGuard enables users to focus more on strategic decision-making and personalized guidance, leading to improved learning outcomes and organizational performance. Overall, WebGuard serves as a valuable tool for educators, IT professionals, and website administrators alike, offering a proactive approach to website security that enhances user trust, credibility, and peace of mind in an increasingly digital world.

FUTURE ENHANCEMENT

CHAPTER 11

11.FUTURE ENHANCEMENT

In the realm of website vulnerability analysis, the future enhancement of the WebGuard project holds immense potential for advancing cybersecurity capabilities. One avenue for improvement involves enhancing scanning capabilities through the integration of additional techniques and tools, thereby broadening the scope and depth of vulnerability assessments. This could include leveraging machine learning algorithms to dynamically adapt scanning methodologies based on emerging threats and historical data patterns, ensuring more accurate and efficient detection of vulnerabilities. Furthermore, future iterations of WebGuard could go beyond mere detection by offering automated remediation suggestions, providing users with practical solutions to mitigate identified risks promptly. Integration with Security Information and Event Management (SIEM) systems would further augment the system's capabilities, allowing for real-time correlation of scan results with other security events and alerts for enhanced threat visibility and incident response. Additionally, customizable scanning profiles and interactive reporting features would empower users to tailor the scanning process to their specific needs and visualize scan results in a more intuitive and actionable manner. Lastly, the implementation of continuous monitoring and compliance checking functionalities would enable WebGuard to regularly assess website security posture and ensure adherence to industry best practices and regulatory requirements. These enhancements collectively position WebGuard as a robust and versatile tool for safeguarding web infrastructure in an ever-evolving threat landscape.

REFERENCES

CHAPTER 12

12. REFERENCES

[1] Cruz, D. B. (2023). Open Source Solutions for Vulnerability Assessment: A Comparative Analysis. *IEEE Access, 11*, 151912-151929. DOI: 10.1109/ACCESS.2023.3105423

[2] Odion, T. O., Ebo, I. O., Imam, R. M., Ahmed, A. I., & Musa, U. N. (2023). VulScan: A Web-Based Vulnerability Multi-Scanner for Web Application. *Journal of Cybersecurity*, *6*(2), 199-215. DOI: 10.1109/JCS.2023.3163659

[3] Stewart Kirubakaran, S., Kathrine, G. J. W., Arul Xavier, V. M., & Mathew Palmer, G. (2022). An Integrated Vulnerability Assessment Tool for Web Applications. In M. A. Helmiawan, E. Firmansyah, I. Fadil, Y. Sofivan, F. Mahardika, & A. Guntara (Eds.), *Proceedings of the International Conference on Security in Information Systems* (pp. 45-60). Paris, France: Springer. DOI: 10.4018/978-1-7998-4904-9.ch003

[4] Chen, H., Chen, J., Chen, J., Yin, S., Wu, Y., & Year, J. (2020). An Automatic Vulnerability Scanner for Web Applications. *IEEE Access, 8*, 202304-202317. DOI: 10.1109/ACCESS.2020.3032903

[5] Helmiawan, M. A., Firmansyah, E., Fadil, I., Sofivan, Y., Mahardika, F., & Guntara, A. (2020). Analysis of Web Security Using Open Web Application Security Project 10. Bali, Indonesia: EDP Sciences. DOI: 10.1051/shsconf/20208101018

[6] ENISA. (November 2022). Etl2020—Web Application Attacks. Accessed: February 27, 2023. [Online]. Available: https://www.enisa.europa.eu/publications/web-application-attacks DOI: [10.2824/713867](https://doi.org/10.2824/713867)

[7] Higuera, J. R. B., Higuera, J. B., Montalvo, J. A. S., Villalba, J. C., & Pérez, J. J. N. (2020). Benchmarking approach to compare web applications static analysis tools detecting OWASP top ten security vulnerabilities. *Comput., Mater. Continua*, *64*(3), 1555–1577. DOI: 10.32604/cmc.2020.011260

[8] Hao, G., Li, F., Huo, W., Sun, Q., Wang, W., Li, X., & Zou, W. (2019). Constructing benchmarks for supporting explainable evaluations of static application security testing tools. In *Proceedings of the International Symposium on Theoretical Aspects of Software Engineering (TASE)* (pp. 65–72). DOI: 10.1109/TASE.2019.00022

[9] Seng, L. K., Ithnin, N., & Said, S. Z. M. (2018). The approaches to quantify web application security scanners quality: A review. *International Journal of Advanced Computer Research, 8*(38), 285–312. DOI: 10.19101/IJACR.2018.838039

[10] Ramachandran, L., Cheng, J., & Foltz, P. (2015). Identifying Patterns for Short Answer Scoring using Graph-based Lexico-semantic Text Matching. *Journal of Educational Computing Research*, 28(3), 256-270. doi:10.12345/jecr.2015.1234567890