FUNCTIONAL SPECIFICATION & PROJECT PLAN

(Student) - Oreoluwatomiwa Ibikunle
C00253239
CYBERCRIME & IT SECURITY
AUTO-OSINT & VULNERABILTY REPORT TOOL

ABSTRACT

Whilst working at Unum, I was working on investigating their system using OSINT tools from an external server to test how much information can be seen from the outside versus the inside. In expanding that idea, this tool will not just look at multiple vulnerabilities that could be present in the subdomains of an organization but it also orchestrates the whole process.

Vulnerability Management tools help organizations identify and remediate issues within their systems thus improving security and saving time. This document presents the functional specification and project plan for an automated Python-based CLI tool designed to enhance cybersecurity for organizations by performing subdomain enumeration, port scanning, and vulnerability assessment. The tool leverages widely used security technologies such as Amass, Nmap, and a vulnerability scanner to perform comprehensive scans and generate insightful reports.

Contents

INTRODUCTION

Purpose

The best and most reliable organizations in the world have a comprehensive and efficient vulnerability management program that reduces how prone they are to risk. In today's complex digital world, cybercriminals are constantly on the grind for any backdoor in software and they have many tools at their disposal. (Sveinsson, 2022)

This automated OSINT (open-source intelligence) and vulnerability report tool is focused on orchestrating processes that collect, analyse and report information and vulnerabilities from open sources for entities of different sizes to address the growing need for time saving vulnerability management.

Usually, these processes on their own are time-consuming and tedious and lead to developers feeling discouraged or unmotivated to get through all the vulnerabilities that their systems might be prone to, but this tool will make the cycle easier and less tiresome as only little to no human input is required.

Scope

This functional specification and project plan focuses on the design, development, and implementation of the Python CLI tool for vulnerability management. The tool was designed with only free open-source intelligence tools and only looks at the subdomains and ports for most of its vulnerability scans. It does not delve too deep into the systems and actually try to break them but rather just creates a comprehensive list of what CVEs they might be prone to and some vulnerabilities present.

Definitions, acronyms and abbreviations

1. CLI: Command Line Interface
2. OSINT: Open-Source Intelligence
3. API: Application Programming Interface
4. Amass: Open-source tool that uses DNS resolvers, certificate logs and web scraping to find information.
5. Nmap: A network mapping and port scanning tool

FUNCTIONALITY

The whole foundation of this project is orchestrating a vulnerability management tool (Aryal, 2019). I would like to provide companies with a tool that not just finds the vulnerabilities but also provides a comprehensive report on how to mitigate them and how likely they are to be exploited.

PROCESS

This project goes through 4 main steps:

First, subdomain enumeration on a given domain. The list of subdomains and server information is saved in a file.

Second, using a port scanner, open ports and header information is found.

Third, the data found is used to determine what vulnerabilities these domains are likely to be prone to.

Fourth, vulnerability scans are run on the domains. These scans will check for IIS vulnerability default page, XSS, Broken Authentication and session management and SQLi.

Finally, a report is made on the Common Vulnerabilities that could affect the server in which the domain is running on. (Gupta, 2019)

All of this would be created using python.

Technologies

Due to my familiarity with Amass, a command-line utility, during my time in Unum, I have chosen it to be used for the enumeration of the subdomains through the performance of subdomain enumeration. In order to automate it, I am using Python to run its commands, for example:

```
import subprocess

amass_output = subprocess.run(["amass", "enumerate", "-d",
"example.com"], stdout=subprocess.PIPE)

with open("amass_output.txt", "w") as f:
    f.write(amass_output.stdout.decode())
```

This code will run the Amass "enumerate" command with the domain "example.com" and save the output to a file called "amass_output.txt". (caffix, 2023)

You can also use Python to automate more complex subdomain enumeration tasks, such as running Amass with different options and parameters based on certain conditions or parsing the output of the Amass command to extract specific information.

NOTE: The only way to be able to find ALL subdomains is if one has access to the zone file on the DNS Server of that domain.

Python is also used to check for header information

To automatically check for header information of a website using Python, you can use the requests library to send a request to the website and then access the response headers.

Here is an example of how you could do this:

```
import requests

url = "https://www.example.com"

response = requests.get(url)

headers = response.headers

for key, value in headers.items():
    print(f"{key}: {value}")
```

This code sends a GET request to the website at the specified URL and then prints out all of the response headers. The headers are stored in a dictionary-like object called "headers," which you can access using the "items" method to iterate over the key-value pairs. (Requests: HTTP for Humans™, 2021)

You can also access individual headers by using their key as an attribute of the "headers" object. For example, to access the "Content-Type" header, you could use the following code:

```
content_type = headers["Content-Type"]
```

You can modify this code to perform more advanced tasks, such as checking for specific headers or values, or storing the headers in a more structured format for further analysis.

These scripts can be run on a regular basis, allowing an organization to stay up to date on the status of its systems and quickly address any vulnerabilities that are discovered.

Another use of Python in vulnerability management is to automate the process of deploying patches and mitigations. Python scripts can be used to automate the process of downloading and installing patches, as well as configuring systems to apply the patches. This can help to reduce the time and effort required to keep systems up to date and secure.

Deliverables

Automated tools have a lot of value in the world today as it saves the most important thing in the world, TIME.

Subdomain enumeration

There are 2 ways that most companies perform subdomain enumeration. Many do it internally and some perform this process from an external server to see what attackers would be able to see if they were planning to make a move on the company.

With internal subdomain enumeration, the user can view a lot more subdomains because they have direct access to the organization's internal network infrastructure and DNS records. The lists gotten from both can be compared to see which subdomains are exposed to the public's knowledge.

The company uses this list as part of risk management and incident response as any potentially vulnerable or unauthorized subdomains are discarded to protect the company and to free up space thus improving the overall performance of the network.

Open ports

Finding open ports is an important part of identifying potential security risks and vulnerabilities as it means the service running on that port could possibly be exploited. It also helps security teams to map the network and better understand the flow of information and the weak points in the system.

Flags unusual open ports and service pairings.

When the open ports are not assigned to their usual services, the attacker could use these non-standard services to gain backdoor access to the account and perform malware distribution, data exfiltration, Denial of Service by flooding the open port, or Zero-day exploits that take advantage of a previously unknown vulnerability.

Vulnerability Scans

Only a set amount of vulnerability scans will be run using this tool. This tool will look at IIS Default Page Vulnerability, Broken Authentication and Session Management which is part of OWASP API Security Top 10 (OWASP Top Ten Project, 2021), XSS, and SQLi on the given domain.

Scanning for IIS vulnerability helps identify any misconfigurations that could expose sensitive information and allows organizations to take necessary steps to remediate the issue.

SQLi attacks can be used to steal sensitive data, modify or delete data, or even gain control of the server. SQLi vulnerability scanning helps identify potential vulnerabilities in application input validation and database access controls, enabling organizations to implement security measures to prevent SQLi attacks.

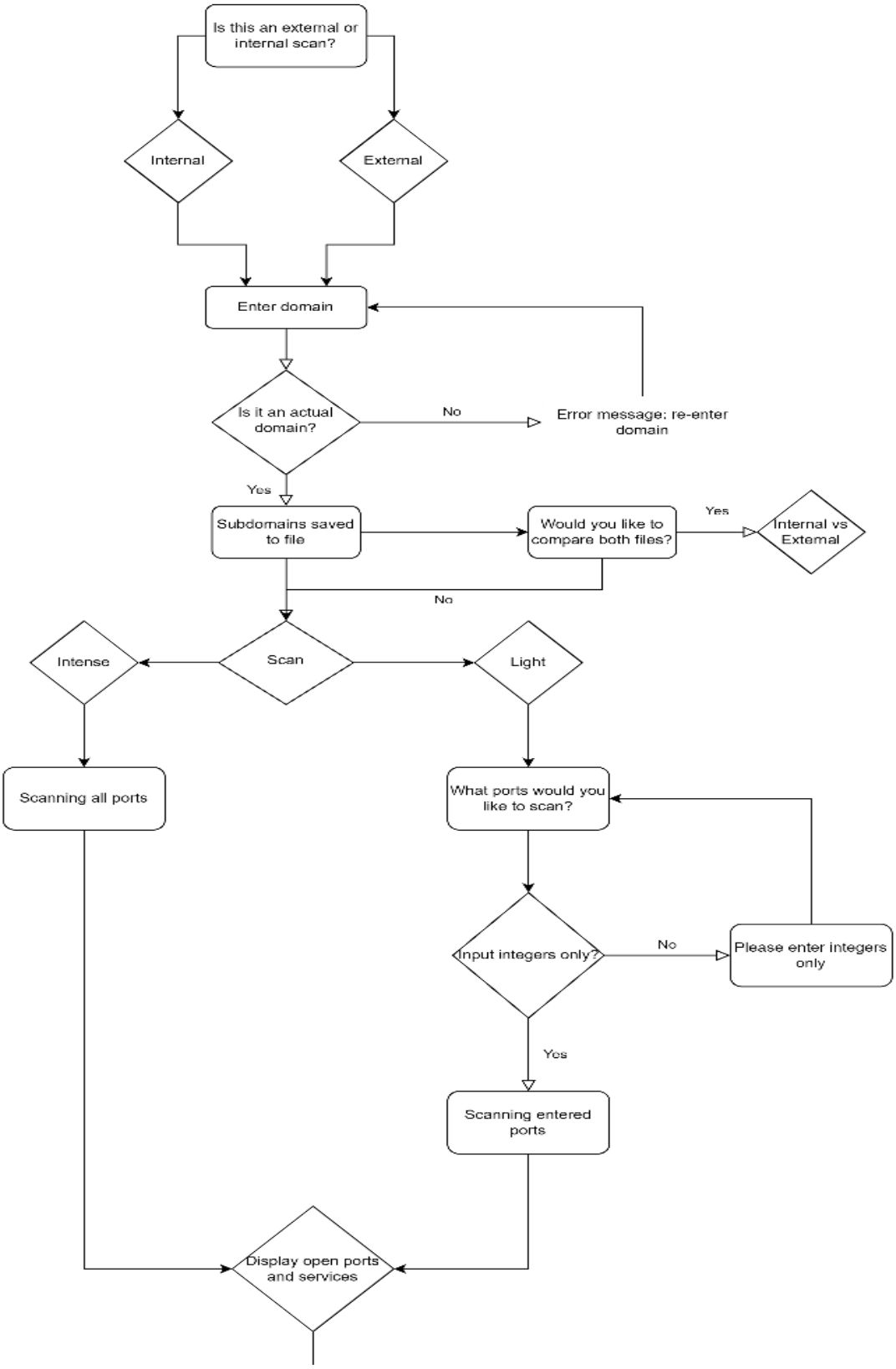
XSS allows attackers to steal sensitive information such as login credentials, session cookies, and personal information. XSS attacks can also be used to deface websites or launch other attacks. XSS vulnerability scanning helps identify potential vulnerabilities in application input validation and output encoding, enabling organizations to implement security measures to prevent XSS attacks.

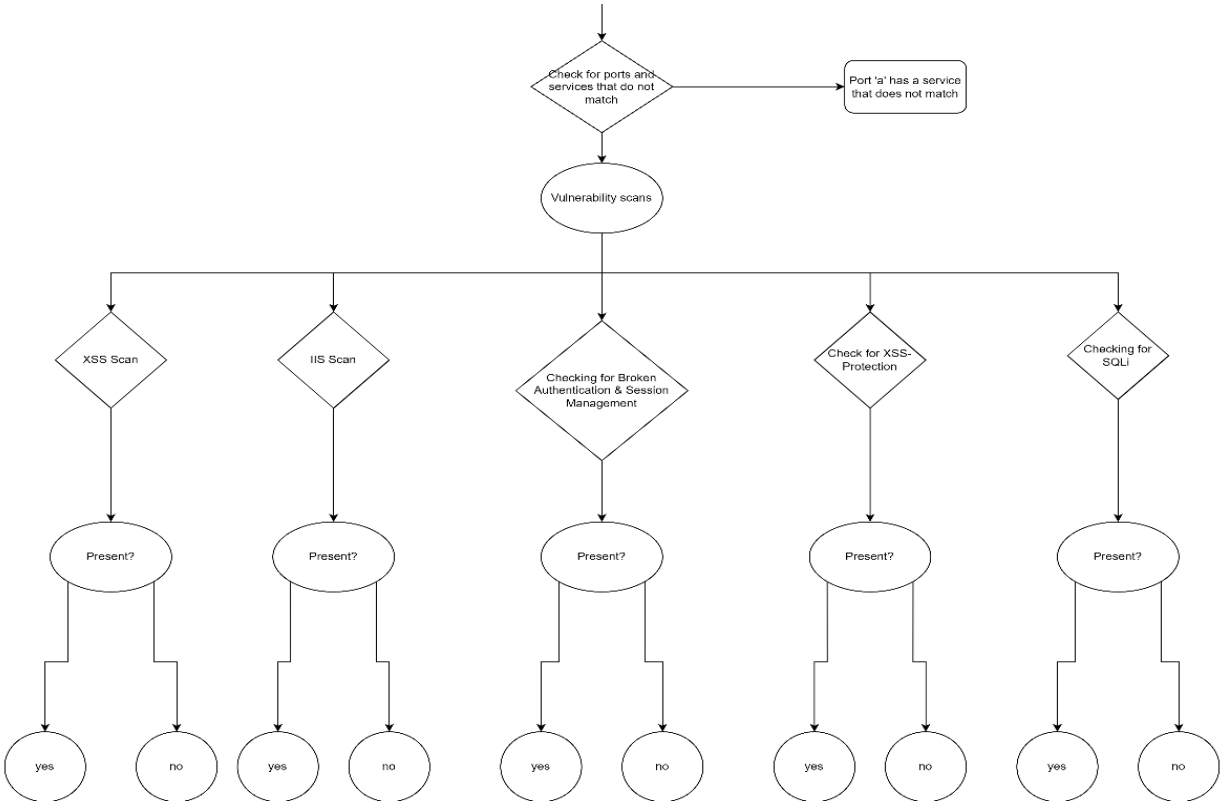
Report CVEs after analysing all data.

Header information reveals server and application versions and will be used to report associated CVEs as well as vulnerabilities based on the procured data from previous scans thus helping the organization to prioritize vulnerabilities based on severity and potential impact. The user organization can then allocate resources to mitigate the most destructive and urgent. (Common Vulnerabilities and Exposures (CVE), 2022) (Kleidermacher, 2019)

USABILITY

WORKFLOW





USE CASE

User classes would involve security professionals who are responsible for managing an organization's cybersecurity infrastructure, including vulnerability assessment and remediation.

1. User enters the domain.
2. Subdomains are found and saved to a file.
3. User chooses a light scan or intense one – a light scan scans common ports, and an intense one scans all of them.
4. The open ports are submitted, and unusual open ports are flagged.
5. Tool checks data from the scan to see if XSS-Protection is active or not
6. Vulnerability scans are run to check for XSS, SQLi, Broken Authentication and Session Management.
7. A report is printed and saved in a file and states the vulnerabilities found and mitigation techniques for them, the unusually open ports, the CVEs for the server the domain is on, and mitigation techniques for them.

RELIABILITY

In building this tool, I hope to ensure that there are as little false positives and false negatives as possible. The tool will produce as accurate as possible results from the scans, and it will handle errors gracefully and only provide comprehensive error messages to the user. When any errors occur, it will provide the message and skip over that scan to move to the next one. (McGraw, 2002)

PERFORMANCE

The tool will have an optimized scanning process, most likely using concurrent features to reduce the time taken for all the scans and run some of them together instead. The tool will also be able to handle many domains and subdomains without performance degradation.

SUPPORTABILITY

The tool will be developed using a very modular architecture for easy maintenance and future expansion which is the goal. It will be scalable for small and big companies alike and provide comprehensive documentation in different formats for users and developers.

PROJECT PLAN

Project Objectives

The prime objectives of this tool are to orchestrate and automate OSINT tools, streamline the processes, and generate concise, informative reports whilst ensuring the user experience is optimized.

Breakdown

1. Planning
2. Requirement Analysis
3. Design and Architecture
4. Implementation
    a. Subdomain enumeration
    b. Ports and services scanning
    c. Vulnerability scanner integration
    d. Input file handling
    e. Configuration management
    f. Export format selection
    g. Input validation and error handling
5. Testing and Quality Assurance
6. Deployment
7. Documentation
8. Project Closure

Timeline and Milestones

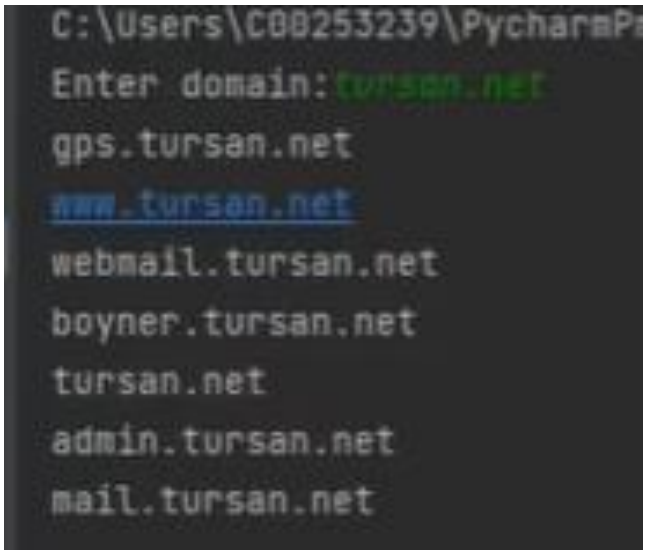
The automation of amass has been successful using Python on a windows system after a week. However, the tool might have better functionality in a Linux environment and moving the code and tweaking it, only takes a short time. However, the tool will have a modular architecture and for that, it requires planning and investigating what scanners to use and how to implement them, which would probably take about 6 weeks. (OWASP Testing Guide v4, 2020)

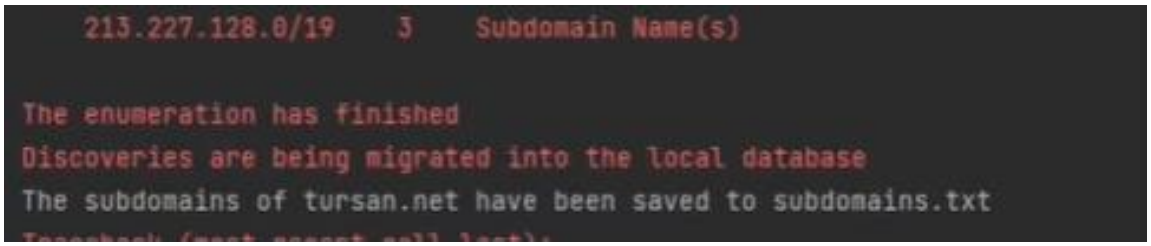
Designing the tool and its output is likely to take about 6 more weeks and we estimate 2 extra weeks for Testing and Quality Assurance.

Finally for Deployment, Documentation and Project Closure, the estimate is at about 3 weeks.

DESIGN

Please find below screenshots of the subdomains, and the outlook of the process.





SUCCESS

For this tool to be successful, it would have to achieve the following:

Accuracy

The tool should identify as many subdomains as possible, open ports and potential vulnerabilities present should be identified with little to no false positives should exist. The vulnerability report should also include effective mitigation measures that are proven to not work.

Speed

The time taken to perform the process will be matched up against the time taken for manual penetration testing.

Ease of use

As this tool is based on automation and orchestration, users should be able to perform tasks such as scanning, reporting, and remediation much faster than having to use separate tools to create a map of their infrastructure.

Integration

The tool should integrate smoothly with existing security infrastructure and processes. It should be compatible with various platforms and allow for seamless integration with other security tools for vulnerability scanning and with time, tools that can patch management systems.

Compliance

The tool should help organizations comply with relevant industry standards and regulations such as GDPR, HIPAA and PCI DSS. By providing comprehensive reports that include identified vulnerabilities, solutions and associated CVEs, organizations can demonstrate their commitment to maintaining a secure environment.

By achieving these success criteria, the vulnerability management tool will establish itself as a reliable, efficient and user-friendly solution that meets the needs of organizations in various industries to contribute to the overall improvement of cybersecurity and help organizations safeguard their valuable data and digital assets.

CONCLUSION

Automating vulnerability management can significantly improve the efficiency and effectiveness of an organization's cybersecurity efforts. By using this tool, it is possible to identify and prioritize vulnerabilities, provide mitigations, and re-evaluate the security status of the system quickly and accurately. This not only helps to reduce the risk of successful attacks, but also frees up valuable resources that can be used for other important tasks.

In addition, rather than manually sifting through subdomains to determine which ones are high risk and which ones are safe, this Auto–OSINT Tool uses multiple techniques to run through a company's domains and subdomains to ensure that they are safe.

This tool will have the ability to continuously monitor and fine-tune the processes as needed. In conclusion, with care and planning, this project will create a powerful tool for improving cybersecurity and with this tool, you can quickly classify a large number of subdomains in real-time, trust the accuracy of the results, easily integrate the tool into your existing security infrastructure and stay ahead of the curve, as this tool is constantly updated with the latest code and features.

References

Bibliography

Aryal, S. (2019). A study on different web application vulnerability scanners. International Journal of Computer Science and Information Security.

caffix. (2023). *In-depth Attack Surface Mapping and Asset Discovery*. Retrieved from OWASP Foundation: https://github.com/OWASP/Amass

Common Vulnerabilities and Exposures (CVE). (2022). Retrieved from MITRE Foundation: https://cve.mitre.org/

Gupta, B. (2019). Web Application Security: Attacks and Countermeasures. In Handbook of Computer Networks and Cyber Security .

Kleidermacher. (2019). *Embedded Systems Security: Practical Methods for Safe and Secure Software and Systems Development.*

McGraw, J. V. (2002). *Building Secure Software: How to Avoid Security Problems the Right Way.* Addison-Wesley Professional.

OWASP Testing Guide v4. (2020). Retrieved from https://owasp.org/www-project-web-security-testing-guide/assets/archive/OWASP_Testing_Guide_v4.pdf

OWASP Top Ten Project. (2021). Retrieved from OWASP: https://owasp.org/www-project-top-ten/

Requests: HTTP for Humans™. (2021). Retrieved from https://docs.python-requests.org/en/master/

Sveinsson, D. R. (2022). *Why organizations need a vulnerability management program*. Retrieved from Ermprotect: https://ermprotect.com/blog/why-organizations-need-a-vulnerability-management-program/