Software Requirement Specification Document for project: Script Generation for Penetration Testing using Artificial Intelligence

Nour Nader, Mohamed Hisham, Mahmoud Osama
Supervised by: Dr. Heba Osama and Eng.Yasmin Kandil

May 3, 2024

Table 1: Document version history

Version	Date	Reason for Change
1.0	25-Oct-2022	SRS First specifications as introduction and system description
1.1	2-Dec-2022	Including functional and Non-functional requirements
1.3	10-Dec-2022	Including project plan and design description and constraints
1.4	11-Dec-2022	Finalizing Appendices and operational scenarios

GitHub:

Contents

Abstract

Penetration testing is the act of a technician with knowledge and experience in network security that will simulate a malicious hacker attack as part of a penetration test to assess the security of a computer network system. As opposed to viruses of a few years ago that would temporarily shut down a system, the effects of modern cyber-attacks can include data theft, ruined networks, and hundreds or maybe millions of dollars in recovery expenses. This project illustrating the practice of developing automated scripts or code snippets that simulate attacks and vulnerabilities in a target system or network serves as an example of script generation for penetration testing. Such scripts' main goal is to accurately represent real-life threat situations, which helps penetration testers assess the target environment's safety measures.

1 Introduction

1.1 Purpose of this document

A Software Requirements Specification (SRS) document is a formal and complete agreement between the client and the development team that outlines the specific requirements for a software system. Its main goal is to give a precise and explicit explanation of what the software is supposed to accomplish, how it should do it, and the limitations that it must work within.

1.2 Scope of this document

A Software Requirements Specification (SRS) document specifies the limits and scope of the software system under development. It describes the features and functionalities that are part of the project as well as, more importantly, what is not. The scope helps in setting clear expectations for both the development team and the stakeholders.

1.3 Business Context

Companies need to generate scripts for artificial intelligence-based penetration testing in order to meet the challenges presented by a continually shifting and complicated cybersecurity environment. Organizations may effectively find and target vulnerabilities thanks to automation powered by AI, which guarantees an accurate evaluation of their security posture. This lowers the price of manual testing and makes it possible to perform security assessments more frequently and rapidly as mentioned here [1]. To stay ahead of cyber opponents, AI tools must be able to adapt to new dangers. Additionally, the increased accuracy of these tools decreases the possibility of missing important vulnerabilities.

2 Similar Systems

2.1 Academic

D Sun, Z Ren, P W Yang, J Li, H Y Chen and T Q Liu suggested that the high error and miss rates of vulnerability scanning equipment make it challenging to meet present needs. When troubleshoot-

ing system vulnerabilities, for instance, the vulnerability scanning equipment merely checks the system version to see if the vulnerability is present. According to reports, their system is still in the development stage, but they expressed confidence that artificial intelligence will advance penetration testing technology in the next years according to[1].

Phillip Garrad and Saritha Unnikrishnan suggested that there is a direct correlation between the number of connected cars on the road and the likelihood of cyberattacks. They say that there is a gap in the application of AI in automotive cybersecurity and that the ideal way to test the potential of reinforcement learning models in CAV security would be to create a simulation environment to assess a few chosen scenarios. as mentioned in[2].

Murat Aydos, Çiğdem Aldan, Evren Coşkun and Alperen Soydan proposed that web applications are being subject to attacks from different locations at various levels of scale and complexity. They said to believe that the results of their study would benefit researchers working on web application security testing. It also could be useful for developers who discuss application security while they develop web applications as learned from.[3]

Zhenguo Hu, Razvan Beuran and Yasuo Tan proposed that penetration testing is done mostly manually and relies heavily on the experience of the ethical hackers that are performing it, called "pentesters". They stated that their framework can already be used to suggest attack strategies to assist with cybersecurity attack training activities. The only pending item of the framework is the Pentetration Tools module, for which implementation is ongoing. Its completion will make it possible to conduct automated attacks on real network environments as mentioned in[4].

2.2 Business Applications

AttackForge [5]: An online platform that offers security teams collaboration features by combining automated penetration testing and vulnerability scanning.

Figure 1: AttackForge

OWASP [6]: ZAP is an open-source security tool that offers automated scanners for different kinds of security testing and is useful for identifying vulnerabilities in web applications.

Figure 2: OWASP

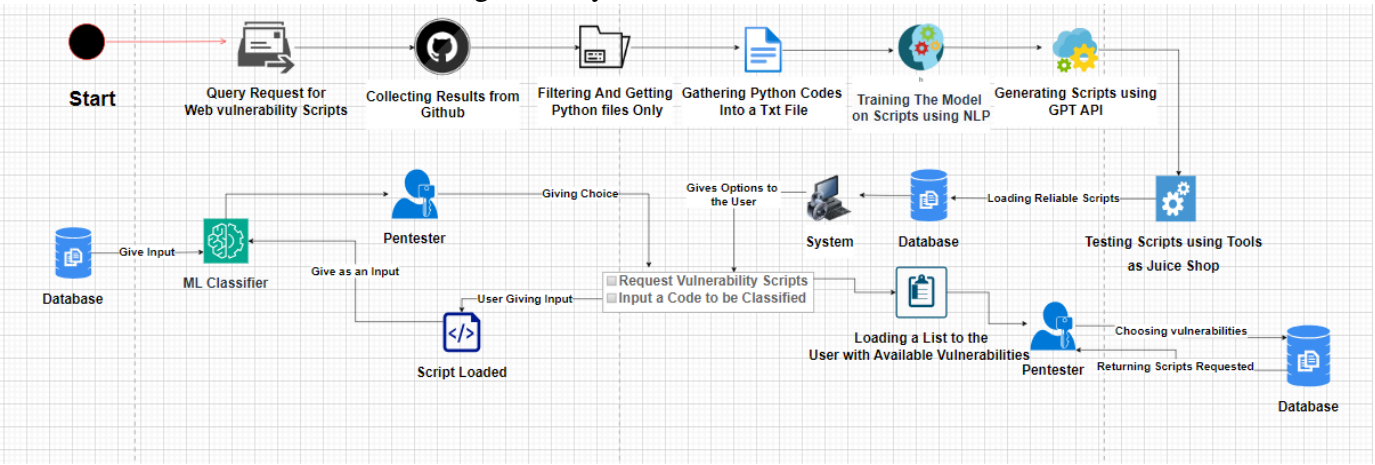
3 System Description

3.1 Problem Statement

The issue with penetration testing is that it requires expert performance and takes a growing amount of time to complete. Also, although manual penetration testing was highly effective at first, it is currently being developed and improved as the need for offensive cybersecurity grows across multiple industries as we found in [7]. Human error, fault-positive scenarios, the vast number of tools that required memorization, the inability to obtain accurate penetration testing, the high cost, and the need for experts and well-trained security professionals to conduct the process which requires a lot of time and training were some of the challenges faced by traditional penetration testing. This highlights the benefits of automating the process, saving time, enhancing the accuracy of the results, and enabling penetration testing by security experts through the use of artificial intelligence as learned in [8].

3.2 System Overview

Figure 3: System Overview



System overall will go as follows:

System will start by issuing a query to github requesting all codes for web vulnerability testing, it then collects all results and filters the files gathering python files only into a txt file. The system Proceeds to train using NLP models on the collected codes making it possible for the GPT API to generate codes for the vulnerabilities. It tests the generated codes using tools as Juice shop and loads the reliable scripts into a database with it's data like description and title. The system gives the user choices whether to request a testing script for a chosen vulnerability or to upload a code and have it classified into a type of vulnerability code. If the user chooses to request scripts, the system gives a list of available vulnerabilities for the user to choose from. The user then gives it's choice and the system returns the requested scripts from the database. If the user chooses to upload a code to have it classified, the system classifies the code using machine learning with the database and code uploaded as an input and returns the classified result to the user.

3.3 System Scope

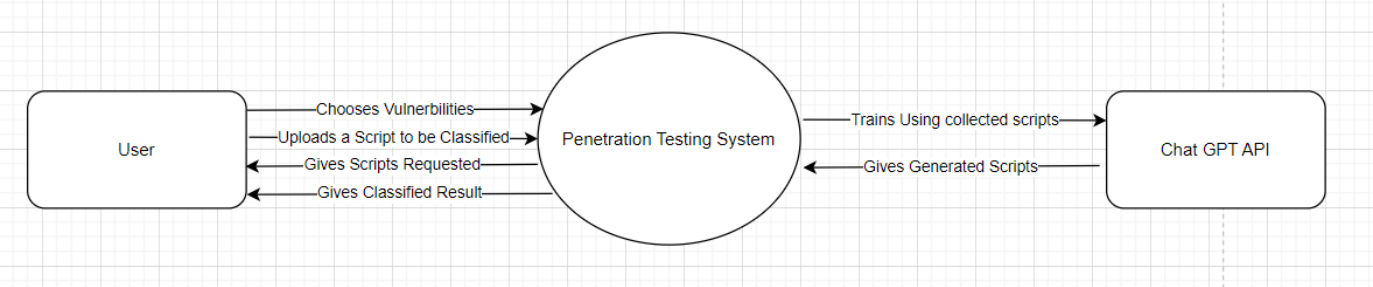
The goal of the Artificial Intelligence The Script Generation for Penetration Testing project is:

- Developing a modern cybersecurity tool aimed at transforming penetration testing processes.
- Creating AI algorithms to discover security weaknesses, write attack scripts, and employ advanced attack methods.
- Providing a scalable, user-friendly solution adaptable to various network environments.
- Incorporating ethical testing processes and employing machine learning for continuous improvement.

3.4 System Context

As shown in the figure, the user will be able to choose from the vulnerabilities and can also upload a script to be classified. The System then can return to the user the scripts requested and the classification results, it also trains the GPT API using the collected scripts. The Chat GPT API can generate scripts and deliver it to the system.

Figure 4: Context Diagram



3.5 Objectives

By examining vulnerabilities, web application security testing tools, and other related topics, this project aims to identify knowledge gaps in web application security testing and fill them. Our goal is to look into and determine the techniques for generating source code with natural language descriptions as the initial investigation in this field of study was started many years ago, but because of technological limitations, the follow-up there haven't been any recent studies.

The main objectives are:

- Creating an advanced cybersecurity solution that automates vulnerability identification, dynamically generates attack scripts based on the weaknesses found, and uses successful attack techniques to simulate real-world cyber threats .
- Ensuring efficiency in a range of patterns by developing adaptability for various network environments. For security professionals to communicate easily, modify testing parameters, and understand findings, a user-friendly interface.
- Scalability in order to handle the growth and complexity of organisational infrastructures. Machine learning integration improves the system's learning capacity over time, and accurate reporting presents results.
- Responsible testing is ensured by ethical considerations, and an organised cybersecurity strategy is encouraged by easy integration with the current security infrastructure.

3.6 User Characteristics

- User should have a good level of knowledge for cyber security.
- User should have a good level of knowledge for coding.
- User should have good understanding of english to be able to use the project.
- User should be an admin of a system or an authorised personal to have permission to use the project.
- User age is important, should be an adult.

4 Functional Requirements

4.1 System Functions

- **ID:1** User should login and confirm that he is an authorised personel.
- **ID:2** User get a list of web vulnerabilities and chooses from them to test on their website.
- **ID:3** User gets a script for requested vulnerability.
- **ID:4** User Uploads a script to be classified.
- **ID:5** User gets a classification result.
- **ID:6** The system can collect codes from repositories.
- **ID:7** The system can train the model.
- **ID:8** The system can test the generated scripts.
- **ID:9** The system can classify scripts.

Figure 5: User's Use case

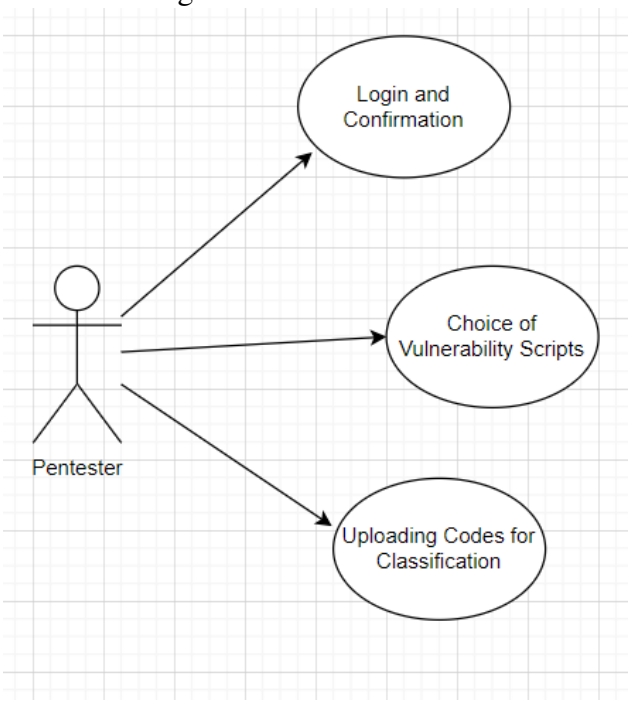
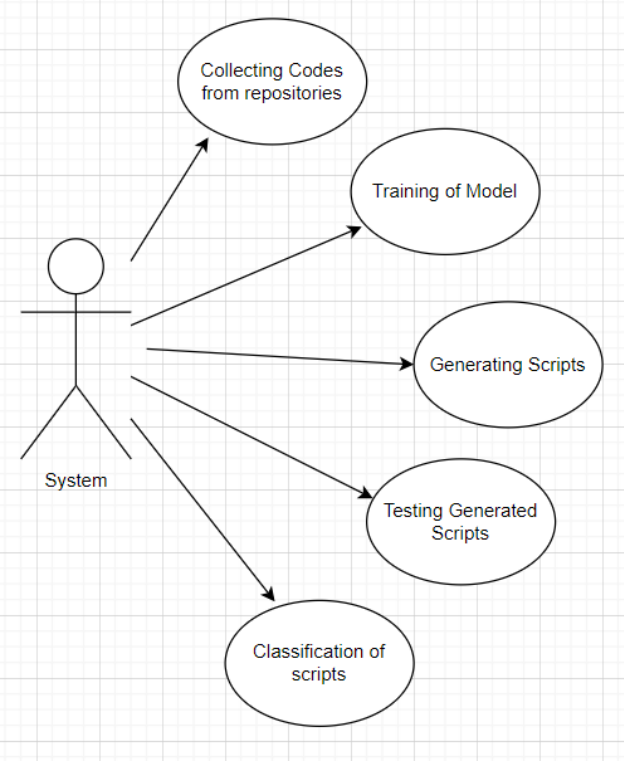


Figure 6: System's Use case



4.2 Detailed Functional Specification

Table 2:

Type	Value
Name	login and confirmation
Code	ID:1
Priority	High
Critical	To obtain security for authorised people only to able to get scripts
Description	User gives information to prove he is an authorised person to gain access
Input	User Data
Output	Access to the system
Pre-Condition	User is authorised
Post-Condition	User gains access
Dependency	Depends on user being a person of authority
Risk	User is not authorised

Table 3:

Type	Value
Name	Choosing of vulnerabilities
Code	ID:2
Priority	High
Critical	To obtain the choice from user
Description	User chooses vulnerabilities to be tested on the website
Input	Vulnerabilities chosen
Output	Scripts from the database
Pre-Condition	Chooses from Web vulnerabilities
Post-Condition	Scripts for chosen vulnerabilites are made
Dependency	Depends on the vulnerabilities available
Risk	..

Table 4:

Type	Value
Name	Script generation
Code	ID:3
Priority	High
Critical	To deliver requested scripts
Description	Generating codes that are specific to requested vulnerabilities
Input	Codes collected
Output	Code generated
Pre-Condition	Vulnerabilites are chosen
Post-Condition	Scripts are genrated and delivered to user
Dependency	Depends on the user request
Risk	scripts being reliable

Table 5:

Type	Value
Name	script Uploading
Code	ID:4
Priority	medium
Critical	To give the user extra help with pentesting
Description	User uploads a code to have it classified into a type of vulnerability.
Input	User's code
Output	Code classification
Pre-Condition	User chooses the option to upload code
Post-Condition	User uploads code successfully
Dependency	Depends on user choosing the option to upload
Risk	User uploading incorrect code

Table 6:

Type	Value
Name	Delivering Classification Result
Code	ID:5
Priority	High
Critical	To give correct classification
Description	A classification for the uploaded is returned to the user
Input	Code uploaded by user
Output	Classification result
Pre-Condition	User uploads a code
Post-Condition	Classification is delivered
Dependency	Depends on the user uploading a script
Risk	Classification is incorrect

5 Design Constraints

5.1 Standards Compliance

This project is made for Web applications only, It can be accessed using a PC or Laptop for the pentester to be able to view Scripts given and give scripts to be classified.

5.2 Hardware Limitations

User needs to have a working laptop or PC to be able to view source code and Upload codes.

5.3 Other Constraints as appropriate

User should be connected to the internet and should have a website to test on.

6 Non-functional Requirements

Non functional requirements in the project are:

6.1 Performance

The project should deliver reliable results and scripts with a minimum error as possible.

6.2 Time

The project shouldn't take much time in the compilation and running phase.

6.3 User-friendly interface

The project should be easy to use and navigate through using a friendly interface.

6.4 Reliability

The project should deliver codes that the user can rely on.

7 Data Design

Data is collected from github repositories then filtered into python codes, model is trained using these codes and generates scripts. Database is then constructed using the reliable scripts generated and tested along with their description and title.

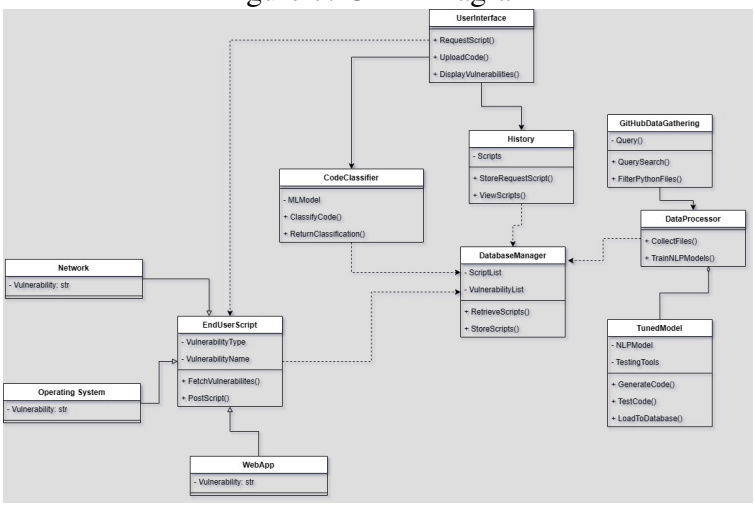
8 Operational Scenarios

Scenario 1: User can gain access when proving he is an authorised person and has the option whether to get scripts for requested vulnerabilities or to upload scripts to be tested.

Scenario 2: User can navigate through the interface and choose from the web vulnerabilities, user then gets the requested scripts.

Scenario 3: User can upload a code to be classifies into a type of vulnerability and get the classification result.

Figure 7: UML Diagram



9 Project Plan

Task	Start date	End date
Preparation	20-8-2023	10-10-2023
Information gathering	12-10-2023	20-10-2023
Collecting information	12-10-2023	24-10-2023
Proposal	9-11-2023	15-11-2023
Train model	23-11-2023	10-12-2023
Researching datasets	23-11-2023	10-12-2023
Ui generation	5-12-2023	10-12-2023
Overview diagram	10-12-2023	10-12-2023
Functional requirements	12-12-2023	10-12-2023
Use cases	12-12-2023	13-12-2023
SRS	18-12-2023	14-1-2024

10 Appendices

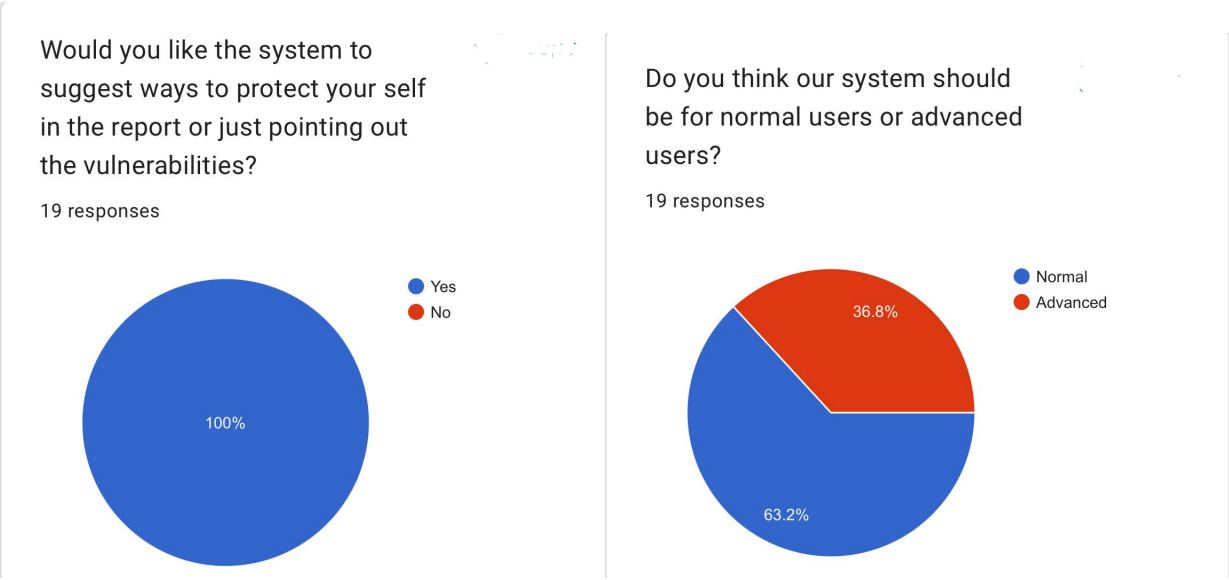
10.1 Definitions, Acronyms, Abbreviations

Table 7:

Name	Description
Vulnerabilities	a software code flaw or bug, system misconfiguration, or some other weakness in the website web application or its components and processes
Scripts	written codes that are interpreted and implemented by a compiler
Penetration Testing	an authorized simulated attack performed on a computer system to evaluate its security
GPT	Generative pre-trained transformer, a type of artificial intelligence language model.
API	stands for Application Programming Interface

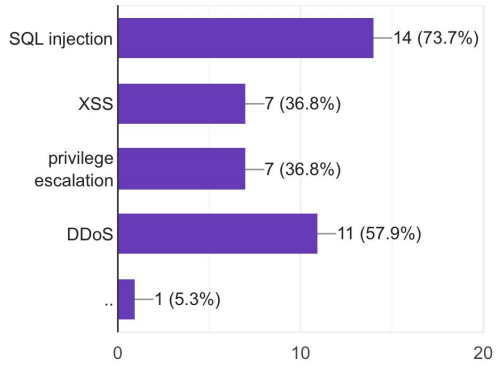
10.2 Supportive Documents

Survey:



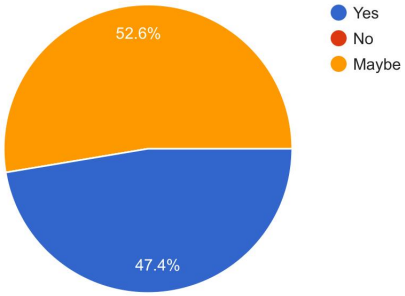
What do you think is the most common security vulnerabilities or attack scenarios that you would like to automate/script?

19 responses



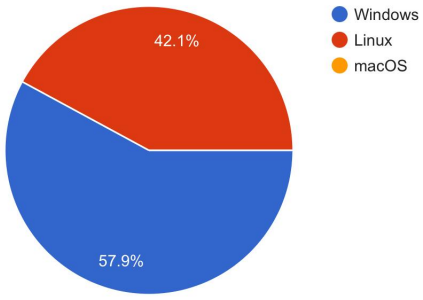
Would you consider integrating AI-generated scripts into your current penetration testing workflow?

19 responses



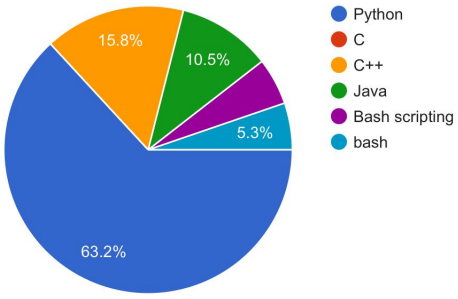
What operating systems are typically targeted during your penetration testing engagements?

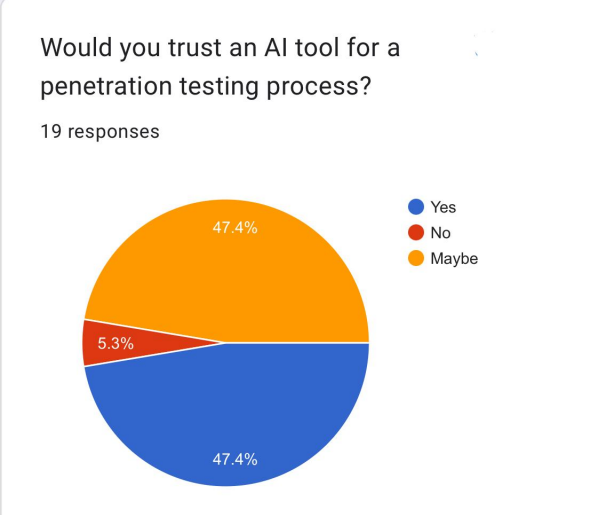
19 responses

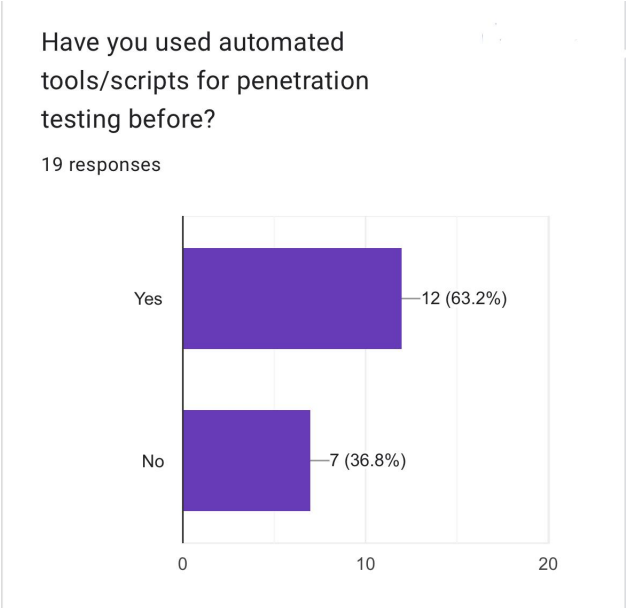


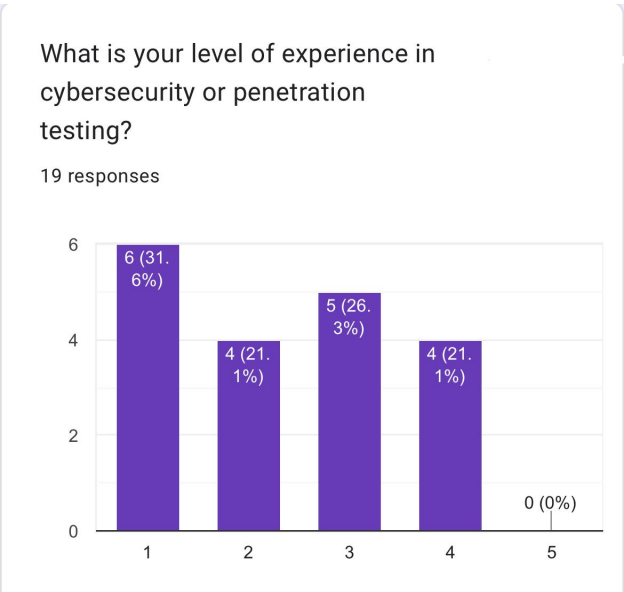
What programming languages are you comfortable with or prefer for scripting tasks?

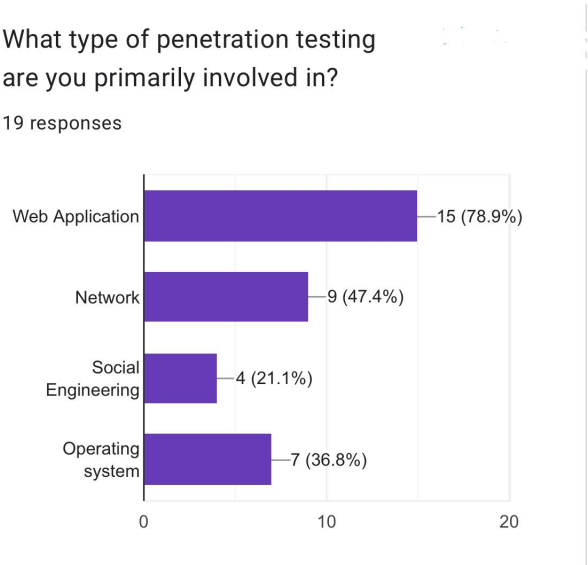
19 responses











Based on our survey we will be mainly focused on the web applications vulnerabilities according to the market needs. Based on results, a project like this is needed and will help the industry, we will take into consideration this information gathered during implementation.

References

[1] D Sun, Z Ren, P W Yang, et al. “Artificial intelligence design research on the cyber security penetration testing of power grid enterprises”. In: *IOP Conference Series: Earth and Environmental Science* (2019).

[2] Phillip Garrad and Saritha Unnikrishnan. “Artificial Intelligence in Penetration Testing of a Connected and Autonomous Vehicle Network”. In: *International Journal of Mechanical and Mechatronics Engineering* (2022).

[3] Murat Aydos and Cigdem Aldan. “Security testing of web applications: A systematic mapping of the literature”. In: *Journal of King Saud University – Computer and Information Sciences* (2021).

[4] Zhenguo Hu, Razvan Beuran, and Yasuo Tan. “Automated Penetration Testing Using Deep Reinforcement Learning”. In: *IEEE* (2020).

[5] *AttackForge*. 2018. URL: `https://attackforge.com/`.

[6] *Owasp*. URL: `https://owasp.org/`.

[7] Sayak Saha Roy, Krishna Vamsi Naragam, and Shirin Nilizadeh. “Generating Phishing Attacks using ChatGPT”. In: *Cornell University* (2023).

[8] Ahmed Bayoumi. “Penetration Testing with AI and Machine Learning”. In: *LinkedIn* (2023).