

IPBeja
Instituto Politécnico de Beja

Escola Superior de Tecnologia e Gestão

CyberForge: A Taxonomy-Driven Framework and Experimental Platform for Experiential Analysis of Web Attack Sub-Vectors

João Carlos Modesto Parra

Dissertation submitted to the Polytechnic Institute of Beja for the award of the Master's degree in computer security in the engineering field.



Instituto Politécnico de Beja
Escola Superior de Tecnologia e Gestão

Beja, July 2026

Instituto Politécnico de Beja
Escola Superior de Tecnologia e Gestão

CyberForge: A Taxonomy-Driven Framework and Experimental Platform for Experiential Analysis of Web Attack Sub-Vectors

João Carlos Modesto Parra

Dissertation submitted to the Polytechnic Institute of Beja
for the award of the Master's degree in computer security
in the engineering field.

Supervisors
Professor PhD Engineer Rui Silva.

Academic Supervisor
Professor PhD Engineer Rui Silva.

Beja, July 2026

Acknowledgements

I would first like to express my deepest gratitude to my wife, whose unwavering support, patience, and understanding accompanied me throughout this demanding journey. During the many months dedicated to study and research, she offered constant encouragement and quiet strength, often placing my academic commitments before her own comfort and time. Her confidence in me and her steadfast presence enabled me to persevere through the challenges of this endeavour. For her companionship, love, and enduring support during this important stage of my life, I am profoundly grateful.

I would also like to extend my sincere appreciation to my academic advisor for their guidance, insight, and encouragement throughout the development of this dissertation. Their expertise and thoughtful feedback were invaluable in shaping the direction and quality of this work.

My gratitude further extends to my family and friends, whose patience, understanding, and encouragement provided constant motivation during this academic journey.

Finally, I would like to acknowledge the professionals and colleagues who generously shared their time, knowledge, and perspectives. Their contributions and constructive discussions enriched the development of this research and helped strengthen its outcome.

Resumo

CyberForge: A Taxonomy-Driven Framework and Experimental Platform for Experiential Analysis of Web Attack Sub-Vectors

As aplicações Web integram os serviços digitais essenciais, no entanto, estão altamente expostas a ataques Web. As categorizações existentes concentram-se principalmente nas vulnerabilidades, e não nos procedimentos operacionais por meio dos quais os atacantes exploram os sistemas. Esta dissertação propõe uma taxonomia coerente de subvetores de ataques Web baseada em padrões documentados e em referenciais, como o OWASP Top Ten [1]. Uma metodologia qualitativa, baseada na análise documental e em casos de estudo, identifica os mecanismos recorrentes e organiza-os numa categorização comportamental que reflete os fluxos reais de exploração [2]. Para operacionalizar este enquadramento, desenvolve-se o CyberForge, uma plataforma interativa que permite uma experiência controlada com aplicações Web vulneráveis. O sistema adota uma arquitetura de duas máquinas virtuais, que separa os serviços do portal da execução dos cenários, em que as aplicações Flask isoladas operam em portas dedicadas. A gestão de sessões baseada em tokens garante autenticação, autorização, contenção e validade temporal explícita. A avaliação demonstra que o CyberForge fornece simulações realistas, mantendo o isolamento rigoroso e auditável, com a utilização eficiente de recursos e feedback mensurável que sustenta a aprendizagem experiencial e a sensibilização para a segurança informática [3], [4]. O trabalho contribui com um enquadramento analítico estruturado para compreender o comportamento dos ataques web e com uma ferramenta prática de formação que reduz a distância entre a teoria e o mundo real.

Palavras-chave: *Web attack vectors, cybersecurity education, web application security, taxonomy, OWASP, information security.*

Abstract

CyberForge: A Taxonomy-Driven Framework and Experimental Platform for Experiential Analysis of Web Attack Sub-Vectors

Web Applications underpin modern digital services but remain highly exposed to cyber-attacks. Existing classifications largely catalogue vulnerabilities rather than the operational pathways through which attackers exploit them. This dissertation proposes a coherent taxonomy of web attack sub-vectors derived from documented attack patterns and established frameworks such as the OWASP Top Ten [1]. A qualitative methodology based on archival analysis and representative case studies identifies recurring mechanisms. It structures them into a behavioural categorisation that reflects real exploitation workflows rather than isolated weaknesses [2]. To operationalise this framework, the research develops CyberForge, an interactive platform enabling controlled experimentation with vulnerable web applications. The system employs a two-virtual-machine architecture that separates portal services from scenario execution, in which attacks target isolated Flask applications bound to dedicated ports. Token-based session management enforces authentication, authorisation, containment, and explicit expiry, ensuring reproducible and ethically bounded use. Evaluation indicates that CyberForge delivers realistic attack simulations while maintaining strict isolation and auditability. Performance measurements show efficient resource utilisation compared with per-scenario virtual machine provisioning, and the platform provides measurable feedback supporting experiential learning and cybersecurity awareness [3], [4]. The work contributes both a structured analytical framework for understanding web attack behaviour and a practical training system that bridges the gap between theoretical knowledge and real-world exploitation dynamics, enhancing cybersecurity competence in academic and professional contexts.

Keywords: *Web attack vectors, cybersecurity training, web security, web application security, taxonomy, OWASP, information security.*

Table of Contents

Acknowledgements v

Resumo vii

Abstract ix

Table of Contents xi

List of Figures xv

List of Tables xix

List of Codes xxi

Acronyms xxiii

1 Introduction 1

1.1 Problem Statement . . . 2

1.2 Research Questions . . . 3

1.3 Research Approach and Methodology . . . 3

1.4 Contributions . . . 4

1.5 Structure of the Dissertation . . . 4

2 Literature Review 7

2.1 Introduction . . . 7

2.2 Web Attacks . . . 8

2.3 Attack Vectors . . . 14

2.4 Categorise the Web Sub-vectors . . . 21

2.5 Literature Review - Research Methodology . . . 37

2.6 Summary . . . 40

3 Attack Vectors 43

3.1 Introduction . . . 43

3.2 Evolution of Web Application Attack Vectors: OWASP Top 10 (2017–2025) . . . 44

3.3 Attack Vector Definitions . . . 46

3.4 Attack Vector Elements 48
3.5 Malware Attack Vectors 50
3.6 The Common Vulnerability Scoring System Framework v4.0 54
3.7 Summary 70

4 Project - CyberForge as an Educational Platform 71
4.1 Introduction 71
4.2 Methodology 73
4.3 Requirement Analysis 77
4.4 Existing Solutions 78
4.5 Proposed Solution 81
4.6 Summary 88

5 Technical Implementation 91
5.1 Introduction 91
5.2 Development of CyberForge 92
5.3 Flask and the CyberForge two-virtual-machine (two-VM) Architecture 99
5.4 Laboratory Environment 100
5.5 Flask Application Structure 101
5.6 Portal Home Page 104
5.7 User Management Module 106
5.8 Scenario Management Module 110
5.9 Progress Monitoring Module 112
5.10 Threat Model Formalisation 113
5.11 Design Trade-off Analysis 115
5.12 Performance Metrics and Measurement Plan 117
5.13 Requirements-to-Control Traceability 118
5.14 Implementation Limitations 118
5.15 Summary 119

6 Evaluation Results **121**

6.1 Introduction . . . 121

6.2 Evaluation Questions . . . 122

6.3 Hypotheses . . . 122

6.4 Functional Validation Results . . . 122

6.5 Security Validation Results . . . 123

6.6 Performance Metrics Evaluation . . . 124

6.7 Traceability Verification . . . 125

6.8 Comparative Evaluation . . . 126

6.9 Validity and Reliability . . . 126

6.10 Pedagogical Observations . . . 126

6.11 Future Work . . . 126

6.12 Summary . . . 127

7 Conclusions and Future Work **129**

7.1 Summary of Findings . . . 129

7.2 Statement of Original Contributions . . . 130

7.3 Design Science Research Perspective . . . 131

7.4 Critical Assessment and Limitations . . . 131

7.5 Future Work . . . 132

7.6 Implications for Cybersecurity Education and Practice . . . 132

7.7 Final Remarks . . . 133

References **134**

Glossaries **149**

Appendices **155**

A Direcotry Structure **158**

List of Figures

Figure 2.1 *Literature Review Map.* 8

Figure 2.2 *OWASP Top 10 2021 draft vulnerability categories.* 9

Figure 2.3 *Revealing the Security-Related Algorithms or Keys on Sina Weibo (V2.4.0_1)* 23

Figure 2.4 *Attack Vector WPA Model.* 25

Figure 2.5 *Design of mobile cloud-based XSS alleviation mechanism.* 29

Figure 2.6 *Kill Chain.* 30

Figure 2.7 *Relations between attack components.* 33

Figure 2.8 *McCumber Cube Taxonomy with Time Element.* 34

Figure 2.9 *Path to Design a Taxonomy.* 36

Figure 2.10 *Data Collection.* 39

Figure 2.11 *NVivo®* Tools. 40

Figure 2.12 *NVivo®* Word Cloud. 40

Figure 3.1 *Evolution of OWASP Top 10 categories across the 2017, 2021, and 2025 releases. The 2025 version introduces new categories and one consolidation, reflecting OWASP's emphasis on root causes rather than isolated symptoms.* 45

Figure 3.2 *Social Engineering is a category of attack.* 47

Figure 3.3 *The social engineering attack vector.* 51

Figure 3.4 *Watering Hole on Facebook.* 52

Figure 3.5 *Botnet attack.* 53

Figure 3.6 *CVSS v4.0 Metric Groups.* 55

Figure 3.7 *CVSS v4.0 Base Score Calculation Flowchart.* 67

Figure 4.1 *Framework for Evaluation in Design Science Research (FEDS).* 74

Figure 4.2 *Complementary nature of design science research and behavioural science research.* 75

Figure 4.3 *Two-VM architecture.* 76

Figure 4.4 *Launch Worflow.* 84

Figure 5.1 *Portal request flow via Nginx reverse proxy and port-bound scenario services.* 103

Figure 5.2 *CyberForge web-based interface Home Page.* 105

Figure 5.3 *CyberForge authentication window.* 106

Figure 5.4 *Layered security enforcement model in CyberForge.* 108

Figure 5.5 *Representative CyberForge database schema supporting users, scenarios, sessions, logs, and scores.* 109

Figure 5.6 *UML-style scenario launch sequence in CyberForge (two-VM architecture).* 111

Figure 5.7 *Data flow diagram for token issuance, validation, and logging/evidence paths in CyberForge.* 112

Figure 5.8 *UML-style module diagram for CyberForge implementation components and their primary data flows.* 113

Figure 6.1 *Launch latency under concurrent load.* 123

Figure 6.2 *CPU utilisation under concurrent load.* 124

Figure A.1 *Directory structure of the Portal VM (VM1).* 158

Figure A.2 *Directory structure of the Scenarios VM (VM2).* 159

List of Tables

Table 2.1 *Web Attacks - Literature Review Summary* 14

Table 2.2 *Attack Vectors - Literature Review Summary* 21

Table 2.3 *Number of attack vectors.* 27

Table 2.4 *Translated attack vectors.* 27

Table 2.5 *Malware Mapped to the Three Pillars of Cybersecurity and Types of Attack Vectors.* 32

Table 2.6 *Related Work - Literature Review Summary* 36

Table 3.1 *Mapping OWASP Categories to CyberForge Scenarios* 46

Table 3.2 *Summary of Azolurt Attack.* 51

Table 3.3 *Summary of Watering Hole Attack.* 52

Table 3.4 *Summary of a Botnet Attack.* 53

Table 3.5 *Example Impact on Score* 59

Table 3.6 *CVSS v4.0 Exploitability Sub-Score* 66

Table 3.7 *Vector String Metrics.* 68

Table 3.8 *CVSS v4.0 Vector Strings with Corresponding Base Scores and Severity Ratings* 69

Table 5.1 *STRIDE threat analysis (threat → impact → mitigation → residual risk) for CyberForge.* 115

Table 5.2 *Design trade-offs in CyberForge implementation.* 116

Table 5.3 *Traceability mapping from requirements to controls and observable evidence within CyberForge.* 118

Table 6.1 *Scenario launch latency under concurrent load.* 122

Table 6.2 *Performance metrics across concurrency levels.* 125

Table 6.3 *Statistical summary of launch latency.* 125

List of Codes

5.1 *Example of a WSGI Application.* 97

5.2 *Example of Flask URL routing for scenario access* 98

5.3 *Example of Rendering a dashboard template using Flask* 99

5.4 *Nginx configuration for the CyberForge portal.* 102

5.5 *UFW configuration rules for the portal VM.* 103

5.6 *UFW configuration rules for the scenario host VM.* 104

Acronyms

AI	Artificial Intelligence 10
APP	Android Application 23
APT	Advanced and Persistent Threats 24, 29, 30, 49, 50
ASA	Cisco Adaptive Security Appliance 61
AV	Attack Vector 54, 56, 58, 59, 61, 62
C&C	Command and Control 52, 53
CF	Collaborative Filtering 10, 11
CGI	Common Gateway Interface 95, 97
CI	Critical Infrastructures 17, 49
CPS	Cyber-Physical Systems 17
CSRF	Cross-Site Request Forgery 76
CT	Combinatorial Testing 26
CTF	capture-the-flag 79–81
CVE	Common Vulnerabilities and Exposures 27, 35, 65
CVSS v2.0	Common Vulnerability Scoring System v2 34
CVSS v3.0	Common Vulnerability Scoring System v3 34
CVSS v3.1	Common Vulnerability Scoring System v3.1 58
CVSS v4.0	Common Vulnerability Scoring System 16–19, 34, 43, 46, 54, 56–67, 69–71
DDoS	Distributed Denial-of-Service 13, 19, 49, 52
DOM	Document Object Model 77
DoS	Denial-of-Service 13, 20, 33, 61
DVWA	Damn Vulnerable Web Application 79, 80

FEDS	Framework for Evaluation in Design Science Research xvi, 74
FGD	Focus Group Discussion 37, 38
FTD	Firepower Threat Defence 61
GA	Genetic Algorithm 11
HIDS	Host-based Intrusion Detection System 11
HTML	Hypertext Markup Language 26–28, 94, 98, 105
HTTP	Hypertext Transfer Protocol 15, 35, 61, 77, 86, 92–96, 99, 102, 103, 105, 112, 123
ICS	Industrial Control Systems 18, 20, 32, 33
IDOR	Insecure Direct Object References 72, 75, 82
IDS	Intrusion Detection System 11, 12, 29
IIS	Internet Information Services 11
IoT	Internet of Things 17, 22, 24, 49, 58
IRC	Internet Relay Chat 19
ISS	Impact Sub-Score 65
IT	Information Technology 32
MVC	Model–View–Controller 105
NIDS	Network-based Intrusion Detection System 11
NIST	National Institute of Standards and Technology 48, 53, 54
NVD	National Vulnerability Database 12, 34, 58
OSCi	operating system command injection 12, 13
OSN	Online Social Networking 27

OWASP	Open Web Application Security Project xv, 1, 7–9, 26, 40, 72, 79, 80
P2P	Peer-to-Peer 19
PII	Personally Identifiable Information 49
QoS	Quality of Service 11
RAID	Redundant Arrays of Inexpensive Disks 19
SCADA	Supervisory Control and Data Acquisition 18, 20
SDN	Software-defined Networks 17
SOAP	Simple Object Access Protocol 11
SQL	Structured Query Language 10–13, 15, 26, 47, 72, 75, 79, 82
SSRF	Server-Side Request Forgery 46
STP	Subtitle Provider Services 23, 24
SUT	System Under Test 26
TPMS	Tire Pressure Monitoring System 22
two-VM	two-virtual-machine xii, 3, 4, 82, 88, 91, 99, 100, 119, 129–131
UFW	Uncomplicated Firewall 102, 103
V2I	Vehicle-to-infrastructure 22
V2V	Vehicle-to-Vehicle 22
VA/SA	Availability Impact 64
VC/SC	Confidentiality Impact 63
VI/SI	Integrity Impact 64
VSD	Variable Speed Drives 17

WA Web Application 1, 7–10, 12–15, 26–28, 30, 34–36, 38

WAF Web Application Firewall 9

WAVSEP Web Application Vulnerability Scanner Evaluation Project 26

WSGI Web Server Gateway Interface 92, 95, 97, 98, 100, 119

XSS Cross-site Scripting xv, 10, 12, 13, 15, 25, 27–29, 72, 76, 79, 82

Chapter 1

Introduction

The pervasive adoption of web technologies has transformed modern society, enabling digital services that underpin commerce, governance, healthcare, education, and critical infrastructure. Web Applications (WAs) now serves as a primary interface between users and complex distributed systems, processing sensitive data and orchestrating essential operations. However, their public accessibility, heterogeneous components, and continuous evolution make them highly attractive targets for cyber attacks. Reports consistently indicate that web application vulnerabilities remain among the most exploited weaknesses in contemporary systems, emphasising the need for robust security analysis and training approaches [5].

Web attacks exploit flaws in software design, implementation, configuration, or user interaction. Prominent examples include injection attacks, cross-site scripting, broken authentication, and access control failures, and widely adopted frameworks such as the OWASP Top Ten catalogue list many of these attacks [5]. While such frameworks provide valuable guidance, they primarily classify vulnerabilities rather than the operational processes through which attackers achieve compromise. In practice, successful attacks often involve multi-stage sequences that combine technical manipulation, protocol misuse, and user interaction. Understanding these processes requires examining attack vectors—the pathways through which adversaries exploit weaknesses to gain unauthorised access or disrupt system behaviour [6].

Attack vectors are rarely isolated. Complex systems allow attackers to chain multiple techniques, forming exploitation workflows that adapt dynamically to defensive measures.

Traditional classifications may therefore obscure the underlying structure of attack behaviour, limiting their usefulness for both analysis and training. A more granular perspective is needed—one that decomposes vectors into meaningful subcomponents that reflect real-world operational patterns. Such a perspective can improve threat modelling, risk assessment, and defensive planning [7].

In parallel, cybersecurity education faces a persistent challenge: bridging the gap between theoretical knowledge and practical competence. Conventional teaching methods often rely on lectures, static examples, or controlled laboratory exercises that do not capture the complexity of real attack scenarios. Research in security education highlights the importance of experiential learning, in which learners actively engage with systems, observe consequences, and reflect on outcomes [8], [9], [10]. However, providing realistic environments for experimentation raises ethical, legal, and safety concerns, as uncontrolled attacks could impact operational systems or violate regulations.

This dissertation addresses these intertwined analytical and educational gaps by investigating web attack techniques from both conceptual and practical perspectives. It proposes a coherent taxonomy of web attack sub-vectors derived from documented exploitation patterns and develops CyberForge, an interactive platform that enables safe experimentation with representative attack scenarios.

1.1 Problem Statement

Despite extensive research on vulnerabilities and defensive mechanisms, no widely accepted framework systematically captures the operational structure of web attacks. Existing taxonomies tend to focus on vulnerability categories or high-level threat types rather than on the detailed pathways through which attacks unfold. Consequently, analysts and learners may struggle to understand how individual weaknesses combine to produce real compromises.

Furthermore, practical training environments capable of demonstrating attack behaviour are limited. Many available tools either oversimplify scenarios or lack adequate isolation mechanisms, reducing their educational value or raising safety concerns. There is therefore a need for both a structured analytical framework and a controlled platform that, together, enable a comprehensive

understanding of web attack behaviour.

1.2 Research Questions

To address this problem, the dissertation presents the following research questions:

1. Can web attack techniques be organised into a coherent taxonomy of sub-vectors that reflects real-world exploitation patterns?
2. Can such a taxonomy be operationalised within an interactive platform that allows users to experience attack scenarios safely and reproducibly?
3. Does a controlled architecture—specifically a separation between portal services and scenario execution—provide sufficient containment for educational use?
4. Can experiential interaction with attack scenarios enhance understanding of web security concepts and promote cybersecurity awareness?

These questions collectively test the central hypothesis that structured analysis combined with controlled experimentation can improve both theoretical and practical cybersecurity competence.

1.3 Research Approach and Methodology

The study adopts a qualitative methodology grounded in archival analysis, document review, and representative case studies. This approach enables the identification of recurring attack mechanisms and the synthesis of common characteristics into a structured taxonomy. The methodology aligns with Design Science Research principles, as it involves the creation and evaluation of a functional artefact intended to address an identified problem [2], [11].

The artefact produced is CyberForge, a web-based educational platform designed to operationalise the proposed taxonomy. CyberForge implements a two-virtual-machine (two-VM) architecture that separates user-facing portal services from vulnerable applications, illustrating

how the system organises these components. Each scenario runs as an independent Flask service bound to a specific network port, while token-based session management enforces authentication, authorisation, containment, and time-limited access. This design ensures realistic interaction with vulnerable systems while preventing unintended propagation of attacks.

1.4 Contributions

The primary contributions of this dissertation are:

- A structured taxonomy of web attack sub-vectors derived from analysis of realistic exploitation patterns.
- The design and implementation of CyberForge, an interactive platform for experiential cybersecurity training.
- A two-VM architecture that isolates portal functions from scenario execution while maintaining operational realism.
- A token-based orchestration mechanism enabling secure and reproducible scenario access.
- An evaluation demonstrating functional correctness, security properties, and educational potential.

Together, these contributions bridge the gap between conceptual threat analysis and practical cybersecurity training.

1.5 Structure of the Dissertation

The remainder of this dissertation proceeds as follows. Chapter 2 reviews the relevant literature on web attacks, attack vectors, and classification approaches, establishing the theoretical foundation for the study. Chapter 3 examines attack vectors in detail, analysing their components and operational characteristics. Chapter 4 introduces CyberForge as an educational platform and presents the proposed architectural solution. Chapter 5 describes the system's technical

implementation, including its modules and deployment environment. Chapter 6 evaluates the platform through functional, security, and performance testing. Finally, Chapter 7 synthesises the findings, discusses limitations, and outlines directions for future research.

This progression reflects the overall research strategy: moving from conceptual analysis to artefact development and empirical evaluation to address both theoretical and practical aspects of web security.

Chapter 2

Literature Review

2.1 Introduction

Chapter 1 introduced the research problem, objectives, and motivation for studying web attack techniques and their impact on modern digital systems. Building on that foundation, this chapter examines the existing body of knowledge on web attacks, attack vectors, and the categorisation of web sub-vectors.

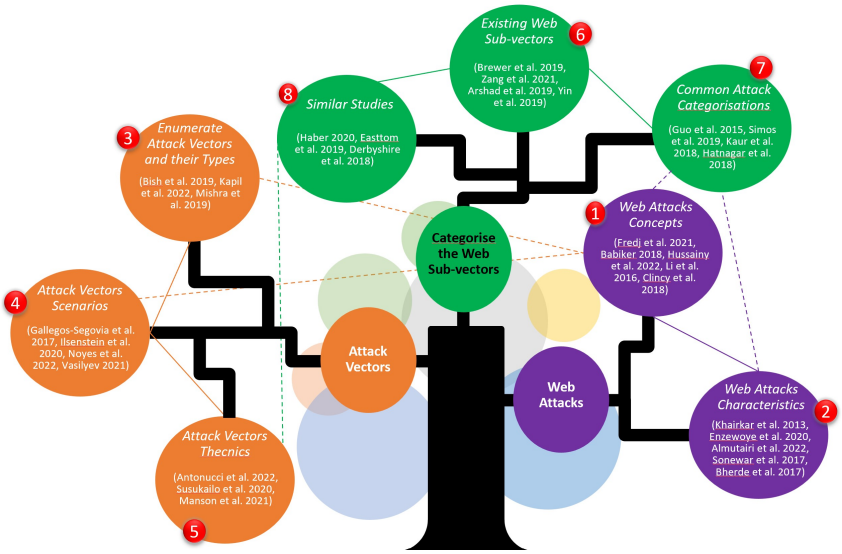
The purpose of this literature review is to synthesise prior research, Figure 2.1, identify established theories and methodologies, and determine gaps that justify the need for further investigation. This work focuses on the characteristics of WA vulnerabilities, common attack techniques, and existing detection and mitigation approaches, including frameworks such as the OWASP Top 10. The chapter also analyses how attack vectors are conceptualised in the literature and explores attempts to classify them into meaningful categories.

By critically evaluating previous work, this chapter establishes the theoretical and conceptual foundations for the study. It highlights limitations in current taxonomies and the lack of practical tools that enable systematic exploration of emerging attack sub-vectors.

While the literature review establishes the theoretical landscape of web attacks and existing classification approaches, a deeper examination of attack vectors themselves is necessary to understand how exploitation occurs in practice. Chapter 3 therefore analyses the structure, characteristics, and operational components of attack vectors in detail.

from Fundamentos de Investigação Cientifica class

Figure 2.1
Literature Review Map.



2.2 Web Attacks

This section describes Web Attacks, the first part of the research guide, as the primary foundation for the study or as influencing it. This section comprises two sub-constructs to cover the main scenarios.

2.2.1 Web Attacks Concepts

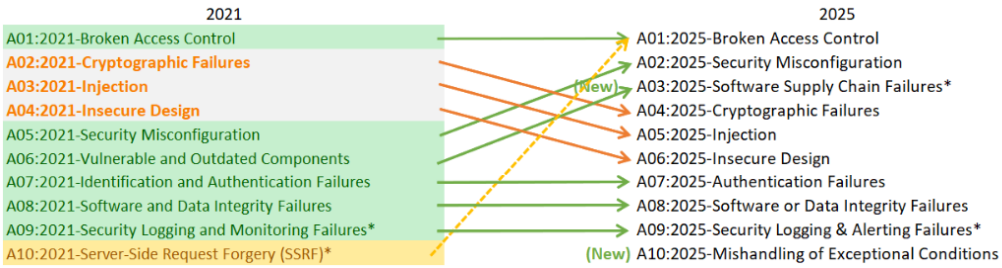
The growing use of the Internet has led to the development of WAs, an essential service for millions of users to obtain information, perform financial transactions, have fun, socialise, and communicate. WAs has become a crucial service in our daily lives and is constantly evolving and being deployed more broadly. Organisations need help developing a WA that meets the requirements for exchanging sensitive data and implementing secure workflows. This study has identified the OWASP's Top 10 (Figure 2.2) most critical web vulnerabilities and corresponding attacks, along with their countermeasures, to ensure that WAs are not subject to the most severe attacks and to prevent previously unknown exploits. It also describes the principal

vulnerabilities and attacks affecting WAs according to the OWASP's Top 10. Their primary purpose in protecting against WAs attacks is the use of Web Application Firewalls (WAFs). However, this study only focuses on WAs as targets, although many other targets are under attack on the Web. They guide us to their literature on server-side web protection methods, including hundreds of works proposing enhanced protection models. They also conclude that existing WAFs employ simple protection rules that cannot predict novel web attack methods, rendering real-time detection impossible. This study provides a comprehensive description of the 10 attacks that could target a WA, along with detailed accounts of countermeasures to mitigate them. A description of web attack concepts will complement our study [1].

Source: OWASP Top 10 – 2021 [12]

Figure 2.2

OWASP Top 10 2021 draft vulnerability categories.



Nowadays, millions of Internet users have daily access to WAs for a wide range of purposes, including e-governments, e-commerce, social networking sites, blogs, content management systems, and webmail, among others. The advanced functionality, ease of access, and availability of WAs have led attackers to launch numerous sophisticated attacks and bypass security controls. The security weaknesses and vulnerabilities of WAs are among the most critical areas in information security and digital forensics. The study examines existing techniques and solutions, including firewalls, intrusion detection systems, honeypots, and forensic techniques, to detect WA attacks. It also explores techniques such as data mining and machine learning to address shortcomings in traditional technologies, aiming to develop more intelligent and convenient methods for detecting WA attacks. In WA forensics, application intrusion detection systems and honeypots have limited real-time detection capabilities, but they achieve high accuracy and performance rates for known attacks. No tests show the effectiveness of data mining in WA

attack forensics. This study reports that 92% of web-based applications are vulnerable, and 75% of all attacks target WAs to compromise information security. They also reported that 70% of web-based attacks succeed, and that WAs experienced up to 27 attacks per minute. This study describes WA attacks, data mining, machine learning and detection techniques. They made a clear, straightforward comparison of detection techniques for WA Attacks and Features. A detailed description of WA attacks will significantly contribute to our study [13].

One of the most critical issues of the Internet is Cyberattacks. Cyberattacks gain unauthorised access to systems, computers, or networks and can damage sectors such as businesses, healthcare, and government. Those cyberattacks could damage computers, disrupt control systems, and cause data breaches, identity theft, phishing, and vandalism. This article focuses on XSS, stored XSS attack types, and Structured Query Language (SQL) Injection because they share similar site management methods, allowing for an in-depth study model to detect them effectively. Artificial Intelligence (AI) analyses large amounts of data at high speeds. When integrated with cybersecurity, security systems can detect and prevent real-time attacks without human intervention, thereby reducing web attacks. The study is limited to two Web Attack types: Stored-XSS and SQL Injection. They focus on a 600% increase in cyberattacks in 2020. They present a detailed and well-developed Proposed Model, with an excellent step-by-step approach from inception, showcasing the progression from code development to tests that demonstrate the achieved results. The contribution to our work supports the two web attacks, XSS and SQL Injection [14].

The constant growth of the Internet, combined with advances in information technology and transmission, has led to the expansion of web services. That web service merger makes it difficult for users to accurately and quickly select suitable services, reducing the utilisation rate of information gathering and sharing. Consumers must understand the safety, capabilities, and functional requirements of WAs and which service best meets their needs. This study examines the Collaborative Filtering (CF) technique used in a web service recommender system. They analyse the vulnerability of the CF technique to web service shilling attacks and to four classical attack models (e.g., average, bandwagon, random, and segment attacks), and examine the impact of distribution-aware Pareto attack models. Experiments are based solely on the user-based CF algorithm, which is a limitation of the study. Web service recommender systems, such as CF,

are widely used for Quality of Service (QoS) management, making them vulnerable to QoS injection attacks. This study does not explore detection approaches for shilling attacks in web service recommendation systems. This paper examines the impact of shilling attack behaviour on CF-based web service recommendations, constructing four basic attack models: average, bandwagon, random, and segmented. They provide us with the basic information for CF-based service recommendations and the algorithms used in their tests, along with an excellent graphical explanation and a detailed description of the tests performed. We are interested in the concepts of four classic attack models explained in this study: average attack, bandwagon attack, random attack, and segment attack [15].

Online stores receive content and data from Web services using the Simple Object Access Protocol (SOAP). The data transmitted between a web service consumer and provider over the network is susceptible to code-injection vulnerabilities, including XPath and SOAP injection. Those attacks can expose valuable information resources. They propose an Intrusion Detection System (IDS) capable of mitigating injection attacks on web services by applying a new attack signature generation method, the Genetic Algorithm (GA). There is no brief discussion of SQL Injection, yet this study uses it. The IDS focuses only on the .NET Internet Information Services (IIS) environment and does not apply to multi-point crossover. Researchers briefly examine XPath Injection and SOAP Injection attacks and propose detection approaches. The evaluation and results are graphically shown and explained. The steps of the GA are well described and easy to understand. A brief discussion of XPath Injection and SOAP Injection attacks will be used for our study [16].

2.2.2 Web Attacks Characteristics

The increasing use of Internet services and applications has also driven e-business and social networking, posing a significant challenge for network administrators and security professionals by exacerbating cyber threats. IDS software or hardware-based monitors network traffic and delivers an alert if it notices malicious activity. This article addresses existing IDS issues, including low false-positive and false-negative rates and data overload, utilising both Host-based Intrusion Detection System (HIDS) and Network-based Intrusion Detection System (NIDS). It discusses

the use of web semantics and ontology concepts to define a methodology for analysing security logs to identify potential security issues and to extract semantic relations between computer attacks and intrusions. The study observes that IDSs cannot effectively protect systems against intruders and their capabilities against three web attacks: XSS, buffer overflow, and SQL injection. The study provides an excellent overview of various security techniques and describes them well. Also, there is a good description of Web attacks and Ontology. This paper includes XSS characteristics, buffer overflow, and SQL injection [17].

The globalisation of WAs enables them to connect to the World Wide Web through web-enabled interfaces, facilitating remote interaction. Web-enabled interfaces enable individuals with varying levels of technical expertise to quickly utilise distinct software components and technologies to develop a WA. This possibility led to an increase in the number of reported software vulnerabilities. This study analyses vulnerability data from software to identify reported WA flaws from 2011 to 2020 and develops an ontology to capture WA weaknesses. Acquired vulnerability data is from the National Vulnerability Database (NVD). This study reveals the need for more automated tools in threat modelling and security requirements engineering. We also noted a limitation on NVD data. The study provides a comprehensive explanation of various web attacks. They also provide detailed descriptions and tables that associate most web attacks with attack methods, protocols, multimedia, languages/frameworks, and configurations/templates. Our interest is the characteristic explanation of several web attacks [18].

Over the past decade, WAs has become a widely accepted method for supporting the Internet. Since then, business requirements have increased, and WA configurations have become more complex. Incorrect input validation and insufficient result validation, often due to poor programming practices or coding errors, make WA a preferred target for Internet intruders. This research will compare vulnerability scanners for WAs that employ dynamic, static, and hybrid approaches. They will scan for SQL injection, operating system command injection (OSCi), path traversal, and XSS web attacks, and select five WAs for scanning based on their PHP and Java features. Although they mentioned the availability of open-source scanners, they used only commercial ones in the study. Dynamic commercial scanners are slower than static ones. Only two of nine scanners effectively detect vulnerabilities in Java. On the other hand, there is a

superb approach to describing SQL Injection, OSCi, path traversal, and XSS web attacks, and a well-done comparison of vulnerability scanner tools, with explanations in tables. The intention is to utilise the excellent characterisation of the web attacks mentioned in [19].

The widespread use of WAs has significantly increased the volume of personal data on the Internet, ranging from simple travel searches to sensitive bank account information. As a result, attackers have become more active in identifying vulnerabilities and developing new web attacks that utilise SQL Injection and XSS. The authors considered a three-tier WAs with static and dynamic behaviour for security and created a mapping model to detect irregularities in SQL Injection and XSS. This study lacks descriptions of static and dynamic WAs and of the vanilla system used for comparison with the proposed method. The study shows that, compared with the vanilla system, the proposed method degrades as request rates reach up to 90%. They clearly describe how SQL Injection and XSS web attacks work. They clearly explain and illustrate the architecture, algorithm, and mapping models of the proposed method for both static and dynamic WAs. The interest refers to SQL Injection and XSS web attacks [20].

The world of Internet WAs has become an essential part of human life, particularly in banking, defence, education, and medical information. Providing secure web services to users is challenging due to a range of attacks. This analysis examines numerous web attacks on web service applications and recent techniques for detecting vulnerabilities in these applications. Contemporary techniques cannot detect all attack types tested in this study and lack sufficient accuracy. The results are based only on the literature review. There is a good list of web attacks in the paper, such as XSS, cross-site request forgery, SQL Injection, Server Misconfiguration, Predictable Page, Breaking Authentication Schemes, Logic, Web of Distrust, XPath injection, XQuery injection, sniffing, request encoding, Distributed Denial-of-Service (DDoS), and Denial-of-Service (DoS). There is a comprehensive survey of approaches to attack detection and prevention, including signature-based, knowledge-based, statistical-based, behaviour-based, and hybrid-based detection. The list of web attacks in the paper will considerably help our study [21].

Table 2.1
Web Attacks - Literature Review Summary

Scholars	Focus	Limitation
(Fredj et al., 2021)	Identify the top 10 web application risks.	Scholars often focus solely on Web applications as targets, although many other targets are also under attack on the Web.
(Babiker, 2018)	It examines several existing techniques and solutions for detecting web application attacks.	It shows that the studied techniques and solutions have limited real-time applicability.
(Hussainy et al., 2022)	Discuss some web attacks.	The researchers investigated only two Web attacks.
(Clincy & Shahriar, 2018)	Propose an Intrusion Detection System.	There is no brief discussion of the attack used in this study.
(Khairkar et al., 2013)	Discuss the use of web semantics and ontology concepts.	The study observed that the Intrusion Detection System could not protect systems efficiently.
(Almutairi et al., 2022)	Compare some vulnerability scanners.	Only two of the nine scanners used are effective.
(Sonewar & Thosar, 2017)	Discuss three-tier web applications.	There are no descriptions.
(Bherde & Pund, 2017)	Analysis of numerous web attacks.	The studied techniques are unable to provide sufficient accuracy for all tested attacks.

2.2.3 Summary

In summary, WAs, which are essential for millions of users to access the Internet, are experiencing rapid growth. As a result, hackers/attackers have increased interest in creating new web attacks.

2.3 Attack Vectors

This section describes Attack Vectors, considered the second part of the research guide, which is required to justify and explore the research problem. Studying existing work at its state-of-the-art level allows identification of current gaps and the direction the study should

take to address the initial problem, thereby generating a hypothesis or proposition. This section comprises three sub-constructs to cover the relevant studies.

2.3.1 Enumerate Attack Vectors and Their Types

Cyberattacks are growing daily, affecting social, economic, and personal lives. In the cloud, many data centres are cloud-enabled and host WAs that deliver services over the Internet. However, open WAs are written in a web programming language and composed of line-by-line code. Therefore, if developers do not properly code the source code of the mentioned applications with respect to security, there is a high probability of an attack on their confidentiality, integrity, and availability, as attackers are always ready to find vulnerabilities in web-based applications by inventing and exploiting new attack vectors. Hence, the top-rated vulnerability in those applications is SQL Injection. This paper proposes enhancing WA security by using a Layer 4 advanced web-based firewall to analyse Hypertext Transfer Protocol (HTTP) request variables through an analyser component. Furthermore, the same firewall has a defence component that protects against injection attacks based on the installed content policy. However, attackers are always ready to exploit vulnerabilities in web-based applications by inventing and executing new attack vectors; an advanced component-based WA firewall will also face these security challenges. Therefore, WA firewalls cannot block or monitor input or output, nor can they enforce policies for specific services, and are configured only for traffic analysis. They talk about the top 10 vulnerabilities highly exploited by attackers, such as SQL injection, broken authentication and authorisation, XSS, broken access control, XML External entities, sensitive data exposure, components with known vulnerabilities, insecure application programming interface known as API, application, and server-level security misconfiguration and insecure deserialisation. Additionally, they present SQL Injection-related attacks from 2012 to 2019, reporting that attacks on WAs increased by 53% from May 2019 to June 2019. There is an excellent description of the various SQL Injection attacks (Error-based, Union-based, Boolean-based blind SQL, and Time-based blind SQL), their attack-vector keywords, and a good description of their proposed architecture. The contributors to our work are the top 10 vulnerabilities, detailed descriptions of the SQL Injection attack types, and keywords related to injection attack vectors

[22].

Internet-based services, an essential part of people's lives, are available online. Invented in 1960 by IBM, virtualisation technology is nowadays, in the digital era, the backbone of cloud technology. The developers created it to optimise the utilisation of physical resources. Virtualisation creates an abstraction layer that separates the different tiers, including data, software, hardware, servers, storage, and networks. This paper discusses the concepts of complete and partial virtualisation, server virtualisation, and desktop virtualisation. The architectures of virtualisation and nested virtualisation are also concerned. Finally, they compare features across several virtualisation tools and present an attack-vector model in a virtualised environment. Unfortunately, there is no comparative performance evaluation of virtualisation tools, and no clarification of security-related issues. A wide range of virtualisation tools is under study, including Xen, VMware ESX/ESXi, Microsoft Hyper-v, Virtual PC, KVM, and VirtualBox. Additionally, there is a comprehensive description of virtualisation technology and architecture, including its pros and cons and various types of virtualisation. Moreover, they provide an excellent illustration of nested virtualisation and of an attack-vector model. Provide clear, expressive graphical examples for all studies. The contribution of this study will be the description of the attack vector model [23].

Today, we witness an unprecedented expansion in cloud computing and cloud services, driven by technological advancements and evolving user needs. Utilising computer resources more efficiently is advantageous for cloud users, enabling them to enhance and pay for services. The major challenge in cloud computing is identifying vulnerabilities and their subsequent exploitation. Identifying cloud-specific vulnerabilities requires an initial examination, followed by the development of a detailed taxonomy based on these vulnerabilities. The basis of their study is on several vulnerability parameters, such as attack vector, Common Vulnerability Scoring System (CVSS v4.0) score, and attack complexity, to propose a comprehensive taxonomy for cloud-specific vulnerabilities. Additionally, they proposed a taxonomy of vulnerabilities to aid in identifying, understanding, and addressing security risks. Unfortunately, the study does not provide a detailed vulnerability analysis to determine the severity score, attack vector, and attack complexity for each vulnerability, making it unclear which vulnerabilities require immediate attention and which experts should handle later. However, this work is well-structured and

grounded in its aim of achieving a comprehensive taxonomy of specific cloud vulnerabilities. In addition, they explain the cloud services model, deployment model, vulnerability assessment model, and cloud-specific vulnerabilities. They also present the public vulnerability repository and taxonomy for cloud-specific vulnerabilities. Finally, it presents the results of the parameters used in the study, including the attack vector, CVSS v4.0 score, and attack complexity. We will utilise vulnerabilities identified in this work and the attack vector list [24].

2.3.2 Attack Vector Scenarios and Characteristics

New tools, such as social networks, email, mobile devices, and Web 2.0 sites, are exposing enterprises to a range of network threats following recent cyberattacks. In addition, new technologies, including virtualisation, mobile devices, the Internet of Things (IoT), and Cloud Computing, have increased the attack surface. As a result, attackers focus their efforts on the weakest link in the information security chain: the users. This paper analyses how attackers achieve their goals, compromise thousands of computers using a combination of technologies, including ransomware manipulation, and generate policies that minimise the risks of information loss by simulating ransomware phases. This study does not analyse the implications of ransomware and other malware attacks on medical devices, the IoT, or Software-defined Networks (SDN). Although the work has an excellent structure and a well-written description of its goal topics, such as Social Engineering, it does not explain what ransomware is or the anatomy of a ransomware attack. Likewise, they clearly outline their implementation and demonstrate a tool for creating ransomware. Finally, they present their analysis results with countermeasures against ransomware. We will use this study as a scenario for an attack vector [25].

As the Internet and internal networks expand, other technologies, such as the IoT and Critical Infrastructuress (CIs), are also growing. The IoT and Cyber-Physical Systems (CPS) comprise a network of interconnected devices, where electrical machines, combined with Variable Speed Drives (VSD), are utilised. The VSD provides those networks with a point of attack on the infrastructure. The paper addresses the cybersecurity of electrical machines by employing a simple switch and additional algorithms to detect cyberattacks via predefined attack vectors. The study focuses solely on a single attack vector that manipulates the frequency-inverter

parameters. The attack vector targeting the system's process is unspecified. However, the aims are well-defined and clearly described. Then, they represent the defined attack vectors of Voltage Detection, Frequency Detection, Phasing Detection, and Unbalance Detection. The Symmetrical Components of a Three-Phase System, Cyber Security Concept, and Results and Outlook are thoroughly explained in the text, complemented by mathematical formulas and graphs that enhance understanding. This study is an excellent complement to the presentation of diverse attack-vector scenarios [26].

Over the past twenty years, smartphones have experienced incredible growth, and today, they are considered small computers that fit in one hand. The globalisation of smartphone use stems from their ability to perform various functions and to enable users to access networks, such as the Internet, to conduct banking transactions, play games, send emails, and engage in e-commerce. Additionally, smartphones store sensitive data, including pictures, call logs, messages, and emails. Therefore, smartphones have become a valuable target for attackers. This paper examines network-based attack vectors on cell phones and discusses several potential attacks stemming from them. They also introduce a novel fuzzing system for network-based attacks. They designed the proposed fuzzing system specifically for testing, ensuring comprehensive evaluation and reliability, and it explores only one of many possible attack vectors on smartphones. However, they demonstrate a structured approach and provide a clear and accurate explanation of the system model and its implementation. It also has a good introduction and background. This study presents additional attack-vector scenarios to enhance the work in [27].

From a political and military perspective, Industrial Control Systems (ICS) are now the most sought-after target for cyberattacks. ICS and Supervisory Control and Data Acquisition (SCADA) systems have become an integral part of technology, playing a vital role in supporting many of our daily activities, such as travel, water and power usage, oil pumping stations, financial transactions, telecommunications, manufacturing, and the operation of telecommunication equipment and communication lines. This study proposes a complex, multi-step modelling approach for attack vectors in a hierarchical fuzzy cognitive map, enabling multiple-scale analysis. They help detail the attack vector during the analysis, enabling a quantitative risk assessment based on the CVSS v4.0 values. This study presents a complex approach to detailing attack vectors, aiming to provide a solution for estimating the local relative risk. Even so, the study is

organised and very detailed. Chapters such as "Cyber Attack Vector Modelling," "Using CVSS v4.0 for Risk Assessment in Modelling Cyber Attacks," and "Example of Modelling the Cyber Attack Vector With A Quantitative Risk Assessment Based on CVSS v4.0" utilise prominent graphs, tables, and descriptions. This study will significantly contribute to our work on attack vector scenarios [28].

2.3.3 Attack Vectors Technics

Modern computing faces an increasingly common threat from botnets. Black-hat hackers and attackers consider several factors when selecting attack vectors. Attackers seek attacks that require substantial resources to invest in and are scalable to the scope they aim to affect. Botnets offers attackers these benefits while remaining efficient and anonymous. This survey examines the drivers of the increased use of botnets, their characteristics, and their attractiveness to attackers. There is a discussion of ways to prevent, detect, and analyse botnet attacks, with a focus on user and system training to prevent attacks and on enhancing security using data gathered from attacks. Short text passages briefly explain the prevention and detection methods, ensuring clarity and accessibility, and there are no experiments or comparisons between modes. The two most important forms of a botnet attack, Internet Relay Chat (IRC) and Peer-to-Peer (P2P), primarily emphasise P2P attacks. This work presents several defence strategies that actively prevent botnet attacks, strengthening cybersecurity measures including monitoring incoming traffic, utilising Redundant Arrays of Inexpensive Disks (RAID), leveraging third-party services, maintaining good internet safety practices, receiving internet security training, employing honeypots, and implementing antivirus software. It also describes various detection strategies, such as artificial neural networks and deep learning, as well as multiple devices that make requests during different network scans and honeypots or honeynets. A report from CloudFlare states that DDoS attempts in the first quarter of 2022 increased by 645% compared to the last quarter of 2021. According to Kaspersky, the botnet Emotet is on the rise again. We will leverage evidence identifying the botnet as a vector [29].

From a cybersecurity perspective, a public health pandemic poses a significant disaster. For example, disasters such as the COVID-19 pandemic have increased the rate of cybercriminal

activity, and consequently, the number of cyberattacks has risen to approximately 230,000 malware incidents per day. During the COVID-19 pandemic, there was a need to analyse a larger number of threats and define security measures to be applied. This article describes the attack methods, vectors, and techniques employed by threat actors worldwide during the COVID-19 pandemic. They also identify common targets of threat actors and cyberattack tactics, and specify possible solutions and improvements in cybersecurity by analysing cybersecurity challenges. This work defines the cybersecurity controls required to secure the analysed attack vectors. This study lacks a conclusion, which is unusual, and it only describes three of an extensive range of attack vectors, each with a brief description. Although brief, the descriptions are clear and understandable and effectively explain the cybersecurity challenges posed by COVID-19, including social engineering attack vectors, the Cyber Kill Chain, attacks via Zoom, and threats to healthcare services. They present potential prevention measures and a table titled "Cybersecurity Toolkit for the Pandemic." A concise yet understandable description of attack vectors will be employed in our study [30].

Special classes of cyber-physical systems include the ICS and SCADA systems, as well as physical industrial systems and facilities. ICS and SCADA are critical infrastructures that support systems and assets essential to companies and industrial operations. Therefore, disrupting their services could have a profound impact on the security, economic well-being, health, and safety of humans or the environment. This study identifies common vulnerabilities and potential attack vectors in critical infrastructures and proposes measures to mitigate these threats. In addition, forensic analysis of an affected ICS device reveals several common attack scenarios and artefacts that aid in diagnosing the attack. Although this study identified several attack vectors, they chose to use only a simple SYN flood DoS attack against an open TCP port. The ICS environment does not encompass the full spectrum of vulnerabilities. However, it provides a detailed description of the background and digital forensics process. The ICS lab, components, methodology, and results are clearly explained in a step-by-step manner, well-supported by figures and graphs. The SYN flood attack description will be used in our work as an attack vector technique [31].

Table 2.2
Attack Vectors - Literature Review Summary

Scholars	Focus	Limitation
(Bisht et al., 2019)	A proposes to enhance web application security.	Firewalls cannot block and monitor input, output, or specific services.
(Kapil et al., 2022)	Discuss the concepts of complete and par virtualisation, server virtualisation, and desktop virtualisation.	There is no clarification on security-related issues.
(Mishra & Singh, 2019)	Propose a comprehensive taxonomy for cloud-specific vulnerabilities.	There is no clarification on security-related issues.
(Gallegos-Segovia et al., 2017)	Discuss ransomware manipulation.	It does not analyse all the ransomware implications.
(Vasilyev et al., 2021)	Create complex multi-step modelling in a hierarchical fuzzy cognitive map.	It illustrates a complex approach to detailing attack vectors.
(Antonucci & Chowdhury, 2022)	Identify the reasons for the increased use of botnets.	They briefly explain prevention and detection methods, and there are no experiments or comparisons between modes.
(Susukailo et al., 2020)	Identify the techniques used by threat actors during the COVID-19 pandemic.	Attack vectors with short descriptions, and there is no conclusion.
(Mason & Zhou, 2021)	Classify common vulnerabilities in critical infrastructures.	Use a modest attack on an open TCP port.

2.3.4 Summary

The interconnection of critical infrastructures with the Internet necessitates advanced technologies; therefore, the growth of the Internet provides attackers with a broader range of attack vectors and the resources to impact critical and other infrastructures.

2.4 Categorise the Web Sub-vectors

This section describes Categorise Web Sub-vectors, which is the third part of the research guide. Researchers recognise this as a unique contribution to the field. Typically, it serves as a research action that influences web attacks and attack vectors. Therefore, this section comprises three sub-constructs to cover the relevant studies.

2.4.1 Existing Web Sub-vectors

The evolution of intelligent devices within the IoT and the exponential growth of electronics have led to a business forecast focused on larger-scale projects, such as smart homes, smart cars, and related applications. For example, a smart car is a complex combination of mechanical systems, electronic components, and intelligent devices. In addition, the growing interest in IoT-connected cars has led to the creation of the "Internet of vehicles" and a transportation environment that includes driverless cars. The scholars evaluated the attack surface, typical vehicle attack vectors, and infrastructure attack vectors. The external connectivity of autonomous cars with the environment and their internal components, such as the Tire Pressure Monitoring System (TPMS), key fobs, and mobile phones, determines the attack vectors. Despite the authors' description of several recommendations for securing vehicles, the study is not robust because it lacks comparisons between the analysed literature and scientific explanations. There are no new or innovative designs in any aspect of security for intelligent vehicles. At the same time, there are no recommendations on the amount of information generated by automotive networks and the IoT. The motivation behind this paper was to review landmark exploits on automobile computer controls and their supporting infrastructure. Automobile automation has become increasingly complex because computers and internal networks control cars, and these systems externally connect to other vehicles through Vehicle-to-Vehicle (V2V) networks. Smart cars are also externally connected to surrounding roadways and traffic control devices in a Vehicle-to-infrastructure (V2I) network environment. Researchers provide an overview of well-known attack vectors in this environment, including "Lock Picking", Vehicle-Monitoring Components, Road Infrastructure, Remote Attacks, and Vehicle Architecture Vulnerabilities. The academics propose a set of recommendations for the secure and safe deployment of smart cars with intelligent infrastructures, including secure sensor design, secure control device design, authenticated and encrypted access to vehicle resources, authenticated and encrypted access to vehicle services, secure road infrastructure design, and paired and encrypted V2I communication. The attack vectors overviewed in this paper will be used in the dissertation [32].

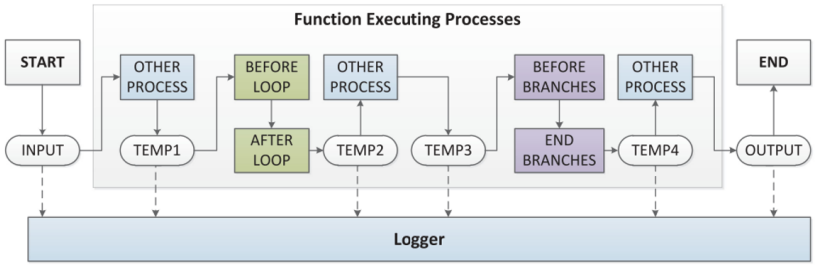
The scholar notes that with the increasing popularity of smartphones, the Android platform

has experienced significant growth. At the same time, Android Applications (APPs) has also risen, making them prime targets for attacks. Network-enabled APPs are vulnerable to Man-in-the-Middle (MITM) attacks, as unprotected related algorithms are susceptible to Reverse Engineering attacks. Although developers improve the security of an APP, they cannot enhance the security of earlier versions, which remain vulnerable to reverse engineering and other attacks. The author explains how attackers exploited poorly protected early versions of APPs as attack vectors, designs a detection strategy, and implements a detection tool that analyses the risks posed by these earlier versions. Zhang describes only two attack vectors. He provided a well-written description of two attack vectors and a corresponding detection strategy. The study's attack vectors are Weak Interfaces and Revealing Security-Related Algorithms or Keys (see Figure 2.3). The detection strategy comprises early-version discovery, Runnable analysis, Vulnerability analysis, and attack analysis. The author implemented an experimental tool called "DroidSkynet". There is an excellent overview and description of DroidSkynet's design. The two Android attack vectors will be presented in detail and will be helpful to the dissertation [33].

Source: 'Looking Back! Using Early Versions of Android Apps as Attack Vectors' [33]

Figure 2.3

Revealing the Security-Related Algorithms or Keys on Sina Weibo (V2.4.0_1)



Along with the growth and popularity of the film industry and the advent of the Internet, web-based databases known as Subtitle Provider Services (STP) make subtitle repositories accessible online for every movie or TV show in various languages. However, security awareness has received insufficient attention despite the widespread use of STPs. Consequently, manipulated subtitle repositories enable attackers to deliver malicious files to users, thereby gaining control of victims' devices, often by exploiting vulnerabilities found in many streaming

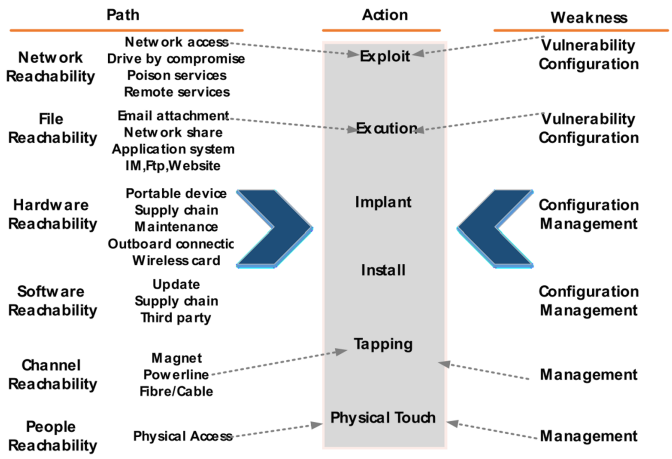
platforms. As a result, attackers are now exploiting subtitle files as a new attack vector for malicious cryptojacking on victim systems, hijacking the victim's resources (e.g., CPU, GPU) to perform cryptocurrency mining without consent. A study of the STP ecosystem reveals that these systems are among the most widespread and readily accessible resources for attackers to exploit. Given the features of the STP ecosystem, it could be considered a new attack vector through subtitle files; however, attackers have not yet fully exploited its potential. This paper fails to specify countermeasures to protect STP ecosystems and to protect users. The scholars describe the STP ecosystem by identifying and analysing its features to assess the current security defence mechanisms and ranking algorithms, demonstrating that the ecosystem is vulnerable and how to manipulate the scoring systems of STPs to make malicious subtitles more popular and, thus, more likely to be chosen by consumers. Primarily, they describe how attackers can exploit vulnerabilities in known media players to launch other attacks via subtitle files. Finally, injects malicious mining code into subtitle files and manipulates the STP 's scoring system to deliver the malicious file, which performs a cryptojacking attack on streaming platforms. This dissertation utilises subtitle files as a new attack vector [34].

Intelligent cities generate a large volume of data and information spanning government affairs, business, daily life, and other domains. This model of cities relies heavily on boundary information technology, including big data, the IoT, and cloud computing. Information leakage poses a significant threat to economic development and social stability if it results in data loss, infrastructure damage, or other security breaches. Intelligent cities and other complex information infrastructures require robust security measures to ensure protection; however, assessing the security risks of such systems has long been challenging, especially in the face of Advanced and Persistent Threatss (APTs). They propose and evaluate an attack vector model based on weakness, path, and action WPA, as shown in Figure 2.4. WPA's proposed model supports the assessment of attack vectors based on both known and unknown weaknesses, depending on the conditions. This work introduces defence factors, proposes an attack-vector evaluation model for integrated defence, and presents an application example of the model. However, the proposed model in this paper cannot predict the likelihood of success for a specific attack vector. The author provides a brief description of attack vectors and conducts a fragile analysis of the evaluation experiment. The scholar proposed a new, innovative attack-vector

evaluation model and a comprehensive method for predicting the availability of attack vectors under defence. The WPA attack vector evaluation model assesses how existing weaknesses affect the likelihood of success, the reachable path, and the feasible actions for an attack vector. Attack vectors, such as network, file, hardware, software, tapping, physical, and action sets, are incorporated into the WPA model. The attack vectors mentioned will be used in our work [35].

Source: 'An Attack Vector Evaluation Method for Smart City Security Protection' [35]

Figure 2.4
Attack Vector WPA Model.



2.4.2 Common Attack Categorisations

With the increasing development of web attack technologies, XSS has been one of the most widespread vulnerabilities in recent years. With XSS vulnerabilities, attackers can steal user accounts and modify essential data. An XSS vulnerability is an injection attack in which attackers send an XSS attack vector to the web server. This paper proposes a method for XSS vulnerability detection based on an optimised attack vector repository that automatically generates an attack vector repository, builds an optimisation model using a machine learning algorithm, and trains it on historical test data generated by the XSS vulnerability detector. This study focuses solely on XSS attacks, and the optimisation model utilises limited training data. The paper describes the framework for XSS vulnerability detection and presents a method for generating XSS attack-vector repertoires, along with an optimisation model. Based on the optimisation model, a

prototype system is implemented and tested with real-world websites. The standard attack types, such as the Hypertext Markup Language (HTML) tag "script", the HTML event attribute, the URL protocol, and the CSS shown in this study, will be utilised in the work [36].

According to several studies, the OWASP Top 10 project identifies command injection attacks as the most common vulnerabilities in software projects. When an attacker attempts to inject arbitrary SQL commands into an application, they use the SQL Injection (SQLi) attack type. SQLi aims to exploit an application database by using SQL tokens in the input, allowing an attacker to inject arbitrary commands into an application's database queries. In addition, it is crucial to provide software tools to filter and sanitise user input. This paper presents a Combinatorial Testing (CT) methodology for WAs to discover SQL Injection vulnerabilities. This work develops and applies three attack grammars to generate concrete attack vectors. Moreover, two distinct Oracle® component mechanisms are presented and evaluated to assess the application's behaviour under these attack vectors. This article does not compare the SQL Injector tool with others, such as Burp Suite and sqlmap. The primary limitation of this experiment was that it primarily focused on WAs with known vulnerabilities to validate its methodology. The scholars applied a grey-box testing technique, as an CT, to discover SQL injection vulnerabilities in WAs for the first time. The CT methodology generates three attack grammar lists: `bool`, `sleep`, and `synerr`. Each row in the above lists produces numerous attack vectors, translating them into concrete SQL command fragments (Table 2.3 and Table 2.4). They developed a Python 3.6 tool, SQL Injector, to exploit attack vectors against a System Under Test (SUT). The mentioned tool incorporates an Oracle® component to determine whether an attack vector resulted in a successful SQL Injection. The academics developed and evaluated two different oracles: the Logging Oracle® and the Syntax Tree Oracle®. They used the Web Application Vulnerability Scanner Evaluation Project (WAVSEP) Framework to evaluate the SQL Injector tool. SQL Injection attack vectors such as `bool`, `sleep`, and `synerr` will be used as common attacks in the dissertation [37].

Source:'Automated combinatorial testing for detecting SQL vulnerabilities in web applications' [37]

Table 2.3
Number of attack vectors.

Sub-grammar	Number of attack vectors		
	$t = 2$	$t = 3$	$t = 4$
`bool`	138	1322	9520
`sleep`	81	569	2873
`synerr`	45	–	–

Source: 'Automated combinatorial testing for detecting SQL vulnerabilities in web applications' [37]

Table 2.4
Translated attack vectors.

Grammar	Attack Vector
`bool` **(t = 4)**	0)%0BAND%0B0%0B=%0Btrue%0B%00%0B a%27)%09anD%090%09=%09!!
`sleep` **(t = 4)**	"(SELECT SLEEP(x)) – +(SELECT SLEEP(x)) #
`synerr` **(t = 2)**	"** %22/**/

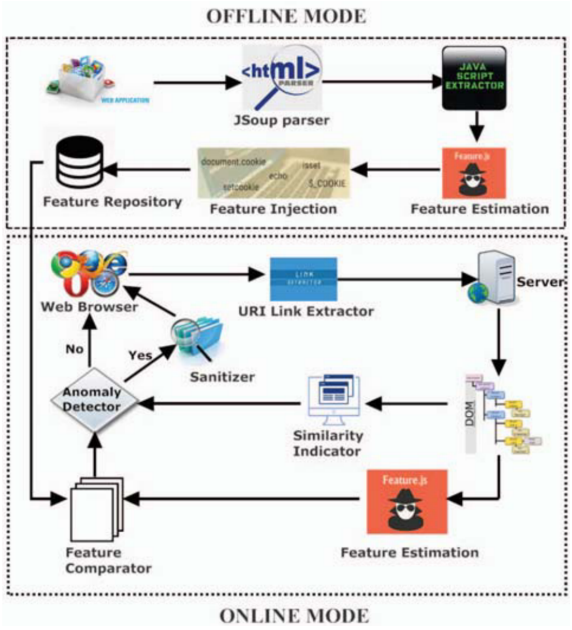
Nowadays, the growth of Online Social Networking (OSN), e-commerce, and banking websites is an indispensable part of human society. Extensively used HTML5 WAs actively address security concerns for both present and future generations. For example, XSS is one of the most commonly used attack vectors, according to the Common Vulnerabilities and Exposures (CVE) repository. In contrast, JavaScript is a channel for meticulously crafted attackers to steal victims' credentials through cyber support. Security teams have implemented several defensive

mechanisms to prevent XSS attacks, but none have focused on determining the JavaScript payload's nested context. This paper presents a sanitisation framework to prevent XSS attacks, including nested context-aware code execution and links, in HTML5 WAs. This framework executes sanitisation procedures on extracted untrusted JavaScript URLs and links, estimating and comparing their features and identifying their diverse nested contexts. Developers designed this study framework prototype for integration as a browser extension; however, it was developed only for Google Chrome and evaluated on virtual platforms rather than in the cloud. This paper proposes a defensive JavaScript architecture that operates in two phases to refine suspicious attack vectors within a nested context. The mentioned phases are online and offline. In the offline phase, the process stores extracted JavaScript code features in a repository after computation. Split into two modules that work together, the first module compares the JavaScript code extracted from external URL links with that extracted from the web page. In contrast, the second module compares the features estimated from the JavaScript code with the features stored in the repository. A sanitised sequence of strings replaces the illicit user's script sequence and determines the nested background of a suspicious sequence of strings in the script (Figure 2.5). From this study, XSS, the top leading vulnerability, will be used in the dissertation [38].

Source: 'Efficient yet Robust Elimination of XSS Attack Vectors from HTML5 Web Applications Hosted on OSN-Based Cloud Platforms' [38]

Figure 2.5

Design of mobile cloud-based XSS alleviation mechanism.



Scholars report that attackers primarily target APTs and financially motivated attacks to send mass spam emails, conduct password hacking, deploy ransomware, exploit vulnerabilities in an organisation's security, and target the human element. To the best of the authors' knowledge, there is an ongoing battle between cybersecurity professionals and attackers, with the attackers generally holding the upper hand. In no time, a skilled attacker can breach a network using multiple attack vectors and entry points to navigate around the defences and evade IDS/Firewall detection. Cybersecurity professionals have the tremendous yet nearly impossible task of securing an organisation's technology, given the proliferation of zero-day vulnerabilities and the lack of secure coding practices. The human element is one of an organisation's greatest strengths and weaknesses. An adequately educated employee is a significant asset to an organisation's defence posture, whereas an untrained one is not. Humans are susceptible to immediately clicking on a malicious URL, browsing suspicious websites, or enabling a script to run in a weaponised document, which remains the number one initial vector for intrusions. Therefore, organisations must be able to prevent and mitigate the risk posed by these attacks. By examining the same tactics

used by malicious actors in a networked environment, this study analyses the approach to APT. The author's approach will utilise the same steps as a real-world attack, including information gathering (footprinting and reconnaissance), intelligence gathering (network scanning), gaining network access, and maintaining access, employing the same tools. This study conducts its experiments in a virtual environment without an Internet connection. There are no rules against APTs using multiple attack vectors, from social engineering employees to dumpster diving to even breaking into the office. This paper illustrates how attackers execute advanced cyberattacks, leveraging tools, methods, and attack vectors to infiltrate a victim's machine, maintain access, and remain undetected within the network. It describes the various stages of an APT, including reconnaissance, intrusion, further capture and persistence, data exfiltration, and attack completion, collectively referred to as the Chain (Figure 2.6). As demonstrated, social engineering, phishing, zero-day vulnerability exploitation, and unpatched browser vulnerabilities are the most common attack vectors in APT attacks, and they will significantly contribute to our dissertation [39].

Source: 'Advance Persistant Threat and Cyber Spying - The Big Picture, Its Tools, Attack Vectors and Countermeasures' [39]

Figure 2.6
Kill Chain.



The author emphasises the importance of WAs as services for millions of users in their daily lives, enabling them to obtain information, perform financial transactions, have fun, socialise, and communicate. Attackers frequently target WAs using attack vectors to gain access and perform malicious activities, such as installing Malware, altering file data, or conducting persistent reconnaissance. It also highlights that threat actors use attack vectors to exploit system vulnerabilities and poor configurations, and introduce stolen credentials to compromise a system. As Haber noted, attack vectors include human elements such as deception and social engineering, as well as physical traits such as fake identification badges. They also include malware, malicious

emails, infected web pages, text messages, and various other forms of deception. The scholar draws our attention to the fact that, despite awareness programs provided by organisations such as MITRE, SANS® Institute, and OWASP, application developers either remain unaware of these vulnerability classes or are unable to implement effective countermeasures. As a result, all methods used by attackers involve intentionally coding software to create a programmatic attack vector (except for social engineering) to exploit a resource for malicious intent. Haber is correct to say that the complexity of the attacks has not changed significantly and that many web problems remain simple, and to pinpoint several technological issues, such as firewalls and endpoint protection solutions. Engineers originally designed those devices to block these attack vectors. Still, in recent years, they have not yet matched the creativity and intent of threat actors, who are more innovative, motivated, and pushing security limits to gain unauthorised access to systems and resources. Haber considers privileged attack vectors, focusing on password hacking and various techniques employed by threat actors to achieve their objectives. Attackers exploit demonstrated techniques to gain administrator privileges when the account already has them, thereby bypassing other security controls, facilitating lateral movement, and opportunistically attempting to crack passwords for privileged accounts on the same or remote systems. As he indicated, password hacking should not be confused with password exposure, such as the use of shared passwords and insecure password documentation. Password hacking is the act of attackers attempting to crack or recover a password using automated techniques. Therefore, academics have primarily focused on privileged attack vectors, one of the three types of attack vectors revealed in this paper. Haber provides an extensive list of password-hacking techniques, including Guessing, Shoulder Surfing, Dictionary Attacks, Brute Force, Pass-the-Hash, Security Questions, Credential Stuffing, Password Spraying, Password Resets, SIM Jacking, and Malware, all thoroughly detailed and explained. Mishra and Singh, in their paper [24], present a taxonomy based on a list of 52 cloud vulnerabilities categorised by attack vectors. In contrast, Haber illustrates attack vectors and their correlation with vulnerabilities. As he states, attack vectors can include Malware (Table 2.5), malicious emails, infected web pages, text messages, social engineering, and various other forms of deception. This book will provide a categorisation of privileged attack vectors, whose work aligns with this vision and provides the foundation for the categorisation and the development of a taxonomy for attack sub-vectors [40].

*Source:*Privileged Attack Vectors [40]

Table 2.5
Malware Mapped to the Three Pillars of Cybersecurity and Types of Attack Vectors.

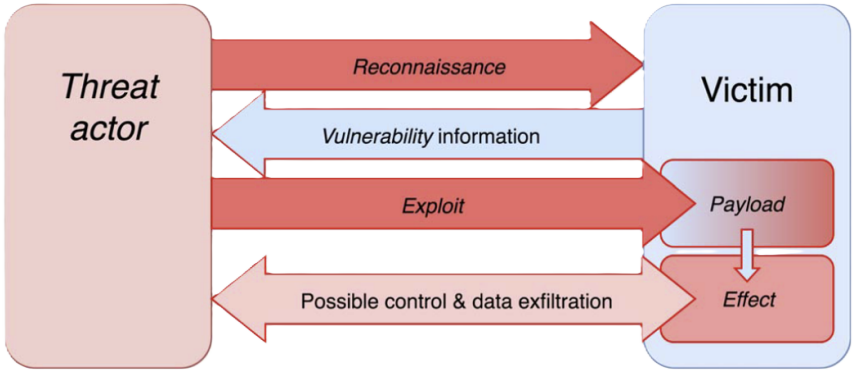
Malware	Privileged Attack Vector	Asset Attack Vector	Identity Attack Vector
Bug	✓	✓	✓
Worms	✓	✓	
Virus	✓	✓	✓
Bots	✓	✓	
Trojan	✓		✓
Ransomware	✓	✓	✓
Adware			✓
Spyware			✓

Information and communication technologies have made modern societies increasingly interconnected, with costs decreasing. Although social media platforms dominate this new era of interconnected systems, they can also aid intelligent living. A prominent example is the automation of critical infrastructure and services such as ICS. Over recent years, we have witnessed several attacks targeting critical national infrastructures. This work combines the analysis of existing components with the creation of defined taxonomies and the application of historical attack data to ICSs, providing a more in-depth understanding of the effectiveness of each taxonomy in categorising varying attacks across diverse infrastructures. This archive does not include taxonomies in the cybersecurity risk assessment process. These taxonomies are effective only in the ICS domains, are not interoperable, and do not facilitate cross-domain communication. Scholars have stated that conventional Information Technology (IT)-based attacks across various ICS environments (Figure 2.7) pose a significant concern not only to system owners and operators but also to society at large. Using taxonomies to categorise cyberattacks

helps us to understand them. Therefore, this paper employs two approaches to evaluate seven taxonomies. Firstly, it analyses different works that define components for creating taxonomies, utilising the set criteria. Secondly, scholars analyse historical attack data to assess the extent to which each taxonomy targets ICSs. Finally, they introduce definitions of attack components to understand the requirements for attack taxonomy within a cyber attack taxonomy, briefly describe the seven taxonomies under review, and evaluate the applied criteria set and historical attack dataset, along with the associated methodologies. The contribution of this paper will be the definitions of attack components, taxonomies under review, and methods for assessing taxonomies [41].

Source:'An Analysis of Cyber Security Attack Taxonomies' [41]

Figure 2.7
Relations between attack components.

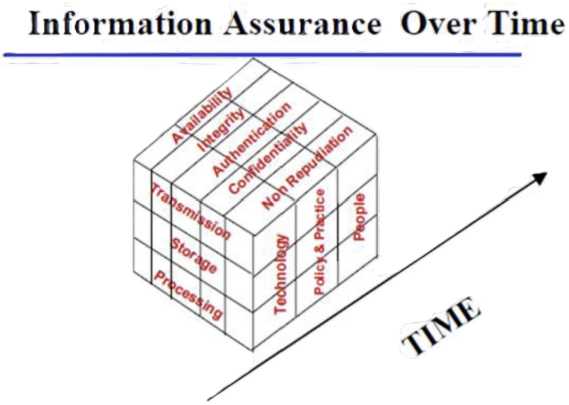


Cyberattacks are not a narrow domain; however, most taxonomies focus on specific malware attacks rather than the broader public domain. For example, many current malware taxonomies emphasise malware behaviour and delivery, dividing Malware into remote access trojans, logic bombs, and ransomware. Although these approaches have been effective in their specific applications, they do not generalise to other forms of cyberattacks and are not necessarily interoperable across contexts. Nevertheless, other attack types require a comprehensive taxonomy, including DoS attacks, attempts to exploit the network via vulnerabilities, social engineering attacks, and other cyberattacks. This paper examines the most current taxonomies that do not encompass all types of cyber-attacks and utilises the McCumber Cube (Figure 2.8) as a foundational methodology for creating an attack taxonomy. Scholars primarily focus on

malware taxonomy and limit its application to the Shamoon virus. They use the McCumber Cube as a methodology to examine security from three dimensions, detailing the McCumber Cube, describing previous attack taxonomies, and proposing new taxonomies. The McCumber Cube will significantly contribute to the dissertation concerning the categorisation of Web sub-vectors [42].

Source:'A Modified McCumber Cube as a Basis for a Taxonomy of Cyber Attacks' [42]

Figure 2.8
McCumber Cube Taxonomy with Time Element.



Black hat hackers gain access to their targets, such as WAs, Internet Infrastructure, LANs, Servers, and other systems, by exploiting vulnerabilities and weaknesses. Therefore, approaches to the attack, such as attack prediction, attack correlation, and vulnerability patching, require knowledge of vulnerability properties, including exploit prerequisites and the impact of exploitation. However, rudeness provides all vulnerability information, not categorised data, from the NVD and vendors. NVD and CVSS v4.0 provide vulnerability descriptions. This paper aims to automate the granular categorisation of vulnerabilities using a Python application. This work only utilises CVSS v4.0 and NVD as vulnerability sources, and the Python application methods require improvement. This study aims to categorise vulnerabilities at a granular level using a Python-developed application. To do so, they utilise vulnerability descriptions from the NVD database, as well as Common Vulnerability Scoring System v2 (CVSS v2.0) and Common Vulnerability Scoring System v3 (CVSS v3.0). They also use the vulnerability properties mentioned above. In addition to the aforementioned properties, scholars introduced two further

properties: probabilistic treatment and the likelihood of successful exploitation. To provide an automated, formalised vulnerability description, they identified three functional properties: impact, prerequisites for vulnerability exploitation, and probability of successful exploitation. The Platform gathers vulnerability information and stores it by CVE ID; categorised vulnerability results are available to everyone, and the community rates each vulnerability, providing a measure of the approach's efficiency and an indication of correctness. A dataset of labelled vulnerabilities is created and made available for general use. The categorisation method and the Python application (`https://github.com/CSIRT-MU/Vulnerability Categorisation`) are used in the dissertation study [43].

In earlier attacks on the Internet, starting from 2007, attackers began by exploiting improper authorisation on Web Servers and WAs. Therefore, the mentioned web attacks become the network's most significant security problems. Unfortunately, existing taxonomies based on attack process, attack classification, and attack category do not effectively categorise attacks by HTTP method. Scholars propose a new taxonomy based on the HTTP method that efficiently returns actions and identifies the characteristic by classifying attacks according to their vulnerability characteristics. The proposed taxonomy focuses on filtering network and computer attacks to classify Web attacks. However, this study does not consider the risks associated with the HTTP method on the client side. It only talks about the dangers of the HTTP method on the server side. This study proposes a new Web attack taxonomy based on dangerous HTTP methods, including `PUT`, `DELETE`, `TRACE`, and `CONNECT`. The taxonomy design considered transmissions between client and server using the HTTP method as mentioned above, resulting in *Client side* → *Operation* → *HTTP method* → *Type of Attack* → *Platform* → *Markup Language* → *Damage* (Figure 2.9). The path to designing a new taxonomy contributes to the purpose of the dissertation study [44].

Source: 'Designing a Taxonomy of Web Attacks' [44]

Figure 2.9
Path to Design a Taxonomy.

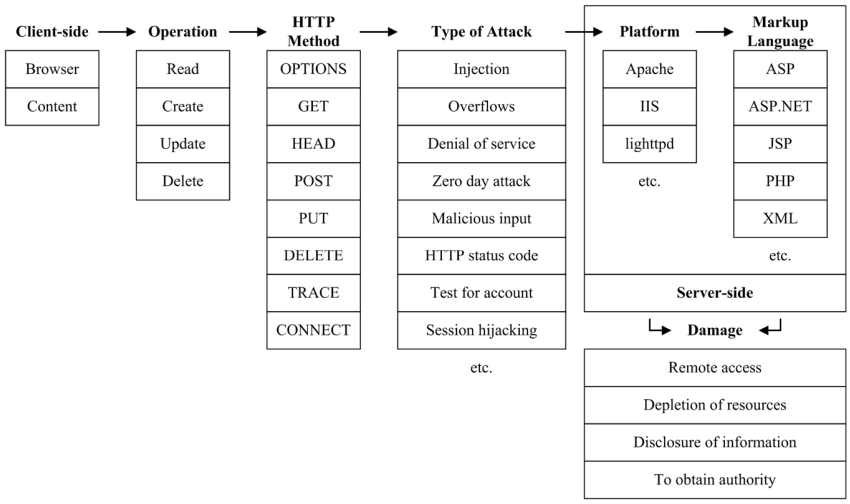


Table 2.6
Related Work - Literature Review Summary

Scholars	Focus	Limitation
(Haber, 2020)	Identify web attack vectors.	Focus only on privileged web attack vectors.
(Easttom & Butler, 2019)	Discuss current taxonomies.	There is no clarification on security-related issues.
(Mishra & Singh, 2019)	Propose a comprehensive taxonomy for cloud-specific vulnerabilities.	Taxonomies with little coverage.
(Bhatnagar et al., 2019)	Discuss the approach to Advanced Persistent Threats.	The test environment lacks the necessary characteristics.
(Yin et al., 2019)	Evaluate an attack vector model.	There is a fragile analysis of the evaluation experiment.
(Simos et al., 2019)	Discuss a combinatorial methodology for web applications.	It is focused only on web applications with known vulnerabilities to validate its methodology.

2.4.3 Summary

As the use of WAs continues to grow, attackers' techniques have evolved rapidly. Unfortunately, current taxonomies do not cover new Web attack vectors. Therefore, a

comprehensive taxonomy of attack vectors and sub-vectors is necessary to explore all dimensions of any attack.

2.5 Literature Review - Research Methodology

The methodological approach employed in the literature review is a qualitative case study methodology, designed to categorise web attack vectors into sub-vectors. This section defines the qualitative case study, explains its application, and justifies the chosen methods. It also outlines the research model, data collection, analysis, validation tools, techniques, and processes to fulfil the study's objectives.

2.5.1 Research Methods

A. **Qualitative Method**

The qualitative method focuses on analysing non-numerical data (e.g., text, video, or audio) to understand concepts, opinions, or experiences [2]. It aims to answer questions such as who, what, where, how, and why. This approach is practical for categorising web attack vectors into sub-vectors by selecting cases from different Web Attack Taxonomies for qualitative analysis. The methodology employs open-ended strategies, including Focus Group Discussions (FGDs), archival analysis, and semi-structured questionnaires [2].

B. **Exploratory Case Study Method**

An exploratory case study investigates contemporary phenomena in their real-life context, particularly when the boundaries between the phenomenon and its context are unclear [45]. This method is suitable for analysing web attack vectors and sub-vectors. The case study components include:

a) **Study Questions**

- Main Question: What sub-vectors should be categorised based on web attack vectors?

- Sub-Questions: What are the main web attack characteristics? What are the attack vector scenarios and their characteristics? How can web attack sub-vectors be categorised?

b) **Study Proposition:** New web attack vector ranges and taxonomies serve as a basis for categorising sub-vectors applied to WAs and Internet infrastructures [2].

c) **Unit of Analysis:** Researchers identify themes by comparing similar case references found in current taxonomies, methodologies, and definitions of attacks [2].

d) **Linking Data to the Proposition:** To achieve triangulation, the researchers employ semi-structured questionnaires, case reference analysis, and archival analysis [45].

e) **Criteria for Interpreting Findings:** The results from the semi-structured questionnaires are analysed [2].

2.5.2 Data Collection Method and Technique

Case studies utilise various sources to collect data (Figure 2.10) and address the five key components of a case study. The primary sources include archival analysis, similar case references, FGDs, and semi-structured questionnaires [2]. The literature review focuses on three types of secondary data:

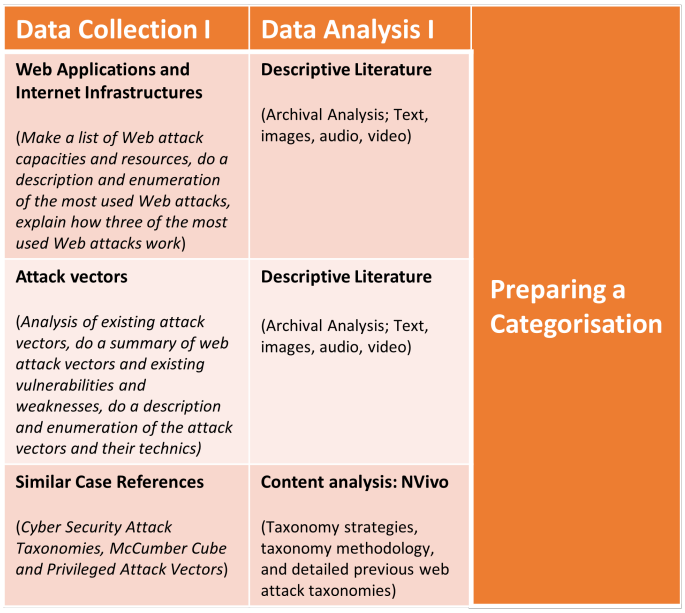
1. **WAs and Internet Infrastructure:** Analyse existing attack vectors, vulnerabilities, weaknesses, and taxonomy strategies [1].

2. **Attack Vectors:** Summarises vulnerabilities and weaknesses, focusing on taxonomy strategies and methodologies [24].

3. **Similar Case References:** Examines Cybersecurity Attack Taxonomies, the McCumber Cube, and Privileged Attack Vectors to develop taxonomy strategies and methodologies [40], [42].

Source: 'DEVELOPMENT OF A FRAMEWORK FOR HERITAGE CONSERVATION MANAGEMENT OF ARAB DISTRICT , ERBIL , IRAQ' [11]

Figure 2.10
Data Collection.



2.5.3 Data Analysis Method and Technique

Robust data analysis involves detailed description, categorisation, and organised processes. Organising the data by thematic relevance enables researchers to conduct a comprehensive analysis and extract meaningful outcomes. *NVivo*® software is employed to support qualitative analysis, reducing manual tasks and enabling researchers to discover trends, recognise themes, and draw conclusions (Figure 2.11). *NVivo*® creates word clouds to summarise similar studies and categorise typical attacks and web sub-vector scenarios as shown in the Figure 2.12.

Source: 'DEVELOPMENT OF A FRAMEWORK FOR HERITAGE CONSERVATION MANAGEMENT OF ARAB DISTRICT , ERBIL , IRAQ' [11]

Figure 2.11
NVivo® Tools.

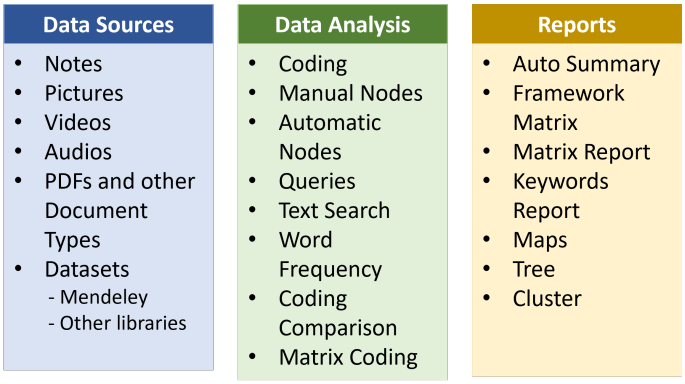


Figure 2.12
NVivo® Word Cloud.



2.6 Summary

This chapter reviewed the existing body of knowledge related to web attacks, attack vectors, and the categorisation of web sub-vectors. It synthesised prior research on common vulnerabilities, threat models, detection techniques, and defensive mechanisms, with particular emphasis on the OWASP Top 10 and related security frameworks.

The analysis highlighted the diversity and evolving nature of attack vectors, demonstrating that modern web threats often combine multiple techniques and exploit complex interactions

among systems. Studies on detection approaches, including intrusion detection systems, machine learning models, and vulnerability scanners, were examined, revealing both their strengths and limitations.

A significant outcome of the review was the identification of gaps in existing work, particularly the lack of comprehensive taxonomies for emerging web attack sub-vectors and the limited availability of practical educational tools for hands-on learning. These findings justified further research and motivated the development of the CyberForge platform.

Overall, this chapter established the theoretical foundation for the study and provided the conceptual basis for the attack vector analysis presented in the next chapter.

Chapter 3

Attack Vectors

3.1 Introduction

The preceding chapter reviewed the literature on web attacks and identified the key concepts, vulnerabilities, and detection approaches that shape the current understanding of web security threats. It also highlighted the need for a deeper analysis of attack vectors and their operational characteristics.

This chapter focuses on attack vectors as the primary mechanisms by which adversaries exploit vulnerabilities in web applications and associated infrastructure. It examines the definitions, components, and structural elements of attack vectors, as well as their manifestations in real-world scenarios, including malware distribution, social engineering, and network-based exploitation techniques.

In addition, the chapter considers methods for assessing the severity and impact of vulnerabilities, notably through the CVSS v4.0. By combining theoretical analysis with illustrative examples, the chapter develops a structured framework for understanding how attack vectors operate and how security analysts systematically evaluate them.

After analysing attack vectors and their underlying mechanisms, this work applies these concepts in an educational context. Chapter 4 introduces CyberForge as a platform for operationalising the identified sub-vectors through interactive experimentation.

3.2 Evolution of Web Application Attack Vectors: OWASP Top 10 (2017--2025)

The structured analysis of web attack vectors requires not only classifying individual vulnerabilities but also understanding how these vulnerabilities evolve in response to changes in technology, development practices, and threat landscapes. The OWASP Top 10 provides a widely recognised and periodically updated reference model that reflects the most critical web application security risks observed in real-world environments. As discussed in the previous chapter, taxonomies of attack vectors benefit from grounding in empirical data and industry-driven classifications [1].

In this context, analysing the evolution between the 2017 and 2025 OWASP Top 10 enables the identification of shifts in attack patterns, the emergence of new categories, and the consolidation of existing vulnerabilities. This comparison supports the refinement of the proposed taxonomy by aligning theoretical categorisation with observed trends in web application security. The structured analysis of web attack vectors requires not only classifying individual vulnerabilities but also understanding how these vulnerabilities evolve in response to changes in technology, development practices, and threat landscapes. The OWASP Top 10 provides a widely recognised and periodically updated reference model that reflects the most critical web application security risks observed in real-world environments [46]. As discussed in the previous chapter, taxonomies of attack vectors benefit from grounding in empirical data and industry-driven classifications [1].

Figure 3.1 illustrates the evolution of the OWASP Top 10 categories across the 2017, 2021, and 2025 releases. The comparison shows that several vulnerability classes persist across versions, while subsequent classifications rename, merge, or reinterpret others to reflect broader architectural, supply-chain, and lifecycle concerns. This progression is particularly relevant to the analysis of attack vectors, since it demonstrates a shift from isolated implementation flaws towards systemic security risks that researchers must consider in both taxonomy development and practical training environments.

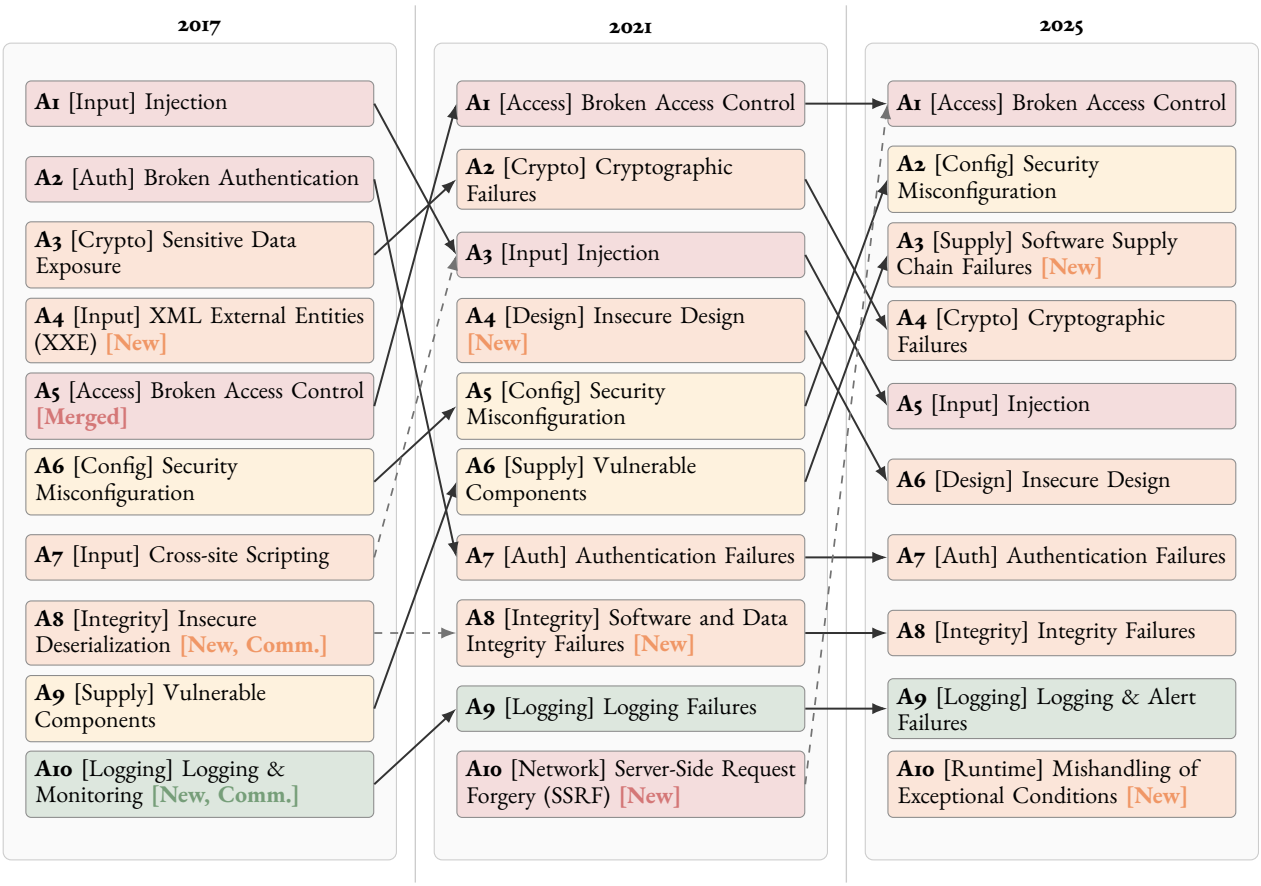
OWASP states that the 2025 release introduces two new categories and one consolidation, while maintaining a focus on root causes rather than isolated symptoms. This shift reinforces

the need to analyse attack vectors not only as individual vulnerabilities but also as manifestations of broader design, configuration, and operational weaknesses [5].

Source: OWASP Top 10 Project Repository [46]

Figure 3.1

Evolution of OWASP Top 10 categories across the 2017, 2021, and 2025 releases. The 2025 version introduces new categories and one consolidation, reflecting OWASP's emphasis on root causes rather than isolated symptoms.



Legend: [Access] authorisation and access control; [Auth] identity and session control; [Input] injection and input handling; [Crypto] data protection and cryptographic mechanisms; [Config] configuration and deployment weaknesses; [Supply] component and supply-chain exposure; [Integrity] integrity of software and trusted state; [Logging] logging, alerting, and monitoring; [Network] back-end request abuse; [Runtime] exceptional-condition handling.

The comparison between the 2017 and 2025 OWASP Top 10 (Figure 3.1) highlights a significant shift in the prevalence of web application vulnerabilities. While earlier classifications emphasised implementation-level flaws such as injection and cross-site scripting, more recent versions prioritise systemic and design-level weaknesses [46]. The prominence of Broken Access Control and the introduction of categories such as Insecure Design reflect a broader focus on

architectural security and secure development lifecycle practices [5].

Furthermore, the continued presence of vulnerabilities in software components and the lack of supply-chain integrity underscore the growing impact of third-party dependencies in modern applications. The emergence of Server-Side Request Forgery (SSRF) and the emphasis on logging and monitoring also indicate heightened concern about back-end interactions and detection capabilities [46]. These trends reinforce the need for a taxonomy that captures both technical vulnerabilities and higher-level systemic risks, aligning closely with the design objectives of CyberForge.

Table 3.1
Mapping OWASP Categories to CyberForge Scenarios

Category	Description	CyberForge Lab Scenario
Broken Access Control	Authorisation bypass	Admin Role Escalation
Broken Access Control	Identity and session control	Vulnerable hash
Injection	SQL/command injection flaws	SQL Injection

By linking standardised vulnerability classifications to executable scenarios (Table 3.1), CyberForge bridges the gap between theoretical taxonomy and practical experimentation. This alignment ensures that the platform remains consistent with industry-recognised security risks while supporting measurable learning outcomes and reproducible training experiences.

This integration of classification and execution represents a key contribution of CyberForge, transforming static vulnerability taxonomies into dynamic, hands-on cybersecurity learning environments.

3.3 Attack Vector Definitions

CVSS v4.0 defines the attack vector as a method attackers use to exploit a vulnerability—that is, the pathway they follow to carry out the attack. CVSS v4.0 uses the attack vector as a metric to assess the proximity or means required to carry out the attack, with higher severity assigned to more remote access by the attacker to the vulnerable system [47].

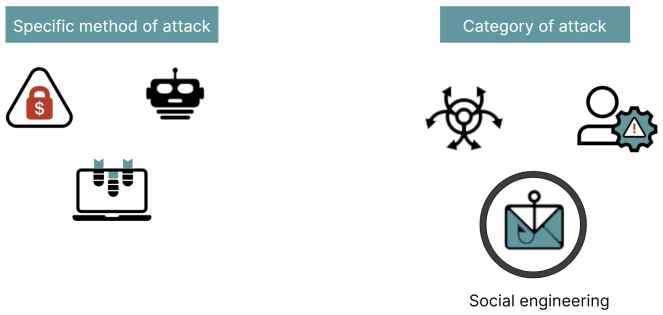
Malik et al. (2023) stated that attack vectors refer to the methods or pathways that cybercriminals use to exploit vulnerabilities and gain unauthorised access to sensitive data. Hackers employ numerous attack vectors to exploit system vulnerabilities, leading to data breaches or the theft of login credentials. The methods involved include sharing malware and viruses, malicious email attachments and web links, pop-up windows, and instant messages that affect the target, as well as duping an employee or individual user [48].

An attack vector is a method by which a threat actor gains access to a system, application, or resource to carry out malicious activities [40].

An attack vector is a method used by a malicious actor to gain unauthorised access to or compromise a network, system, or facility. An Attack Vector can also refer to a specific method of attack or a category of attack (e.g. social engineering) [1], [49].

Source: FCF - Introduction to the Threat Landscape 3.0 [49]

Figure 3.2
Social Engineering is a category of attack.



The specific Injection attack vector enables attackers to inject arbitrary SQL commands, such as queries, thereby executing them in the web application's database via vulnerable web pages [1].

SQL vulnerabilities are prone to attacks that compromise confidentiality, integrity, and availability. Attackers exploit SQL vulnerabilities by injecting and executing various online attack vectors, with success rates of up to 100% [22].

From the definitions above, a possible definition of an attack vector is a pathway, method of attack, or attack category through which a black-hat hacker exploits system vulnerabilities and

misconfigurations, using stolen credentials to compromise the system.

3.4 Attack Vector Elements

An attack vector comprises three elements: a vulnerability, a mechanism (or exploit), and a pathway (or target). To develop effective security measures against potential attacks, it is essential to understand the three crucial elements for identifying and mitigating risks associated with unauthorised access to systems and networks [50].

The attack vector elements, vulnerability, mechanism, and pathway together suggest that they constitute the paths or methods used by a hacker to access a network or computer to exploit system vulnerabilities illegally [50].

3.4.1 Vulnerability

A vulnerability is a weakness or flaw within a system, process, configuration, or protocol or where an attacker could exploit user behaviour to compromise the confidentiality, integrity, or availability of a system or its data, or "*a weakness in an information system, system security procedures, internal controls, or implementation exploitable*" [51], [52]. In practice, vulnerabilities arise from poor coding practices, insecure configurations, Unpatched software, mismanaged privileges, and Human error or lack of security awareness [53], [54], [55].

The vulnerability categories, according to National Institute of Standards and Technology (NIST), include software vulnerabilities (e.g., buffer overflows, SQL injection points, and remote code execution flaws), configuration flaws (e.g., default passwords, exposed admin interfaces, and unsecured ports), design flaws (e.g., flawed authentication logic or weak cryptographic implementations) and human vulnerabilities (e.g., susceptibility to social engineering, poor password practices) [51], [53], [55].

3.4.2 Mechanisms

A mechanism is the method or tool an attacker uses to exploit a vulnerability. An exploit (or mechanism) is a piece of software, a chunk of data, or a sequence of commands that exploits a vulnerability to cause unintended or unanticipated behaviour on a system. An exploit could allow an attacker to execute arbitrary code, gain escalated privileges, or access restricted resources. Exploit development is often employed in penetration testing and red teaming contexts. However, malicious actors also utilise it to build payloads for malware, ransomware, or APT campaigns. The Metasploit Framework is a well-known tool that bundles multiple exploits for testing [51], [55].

The exploit is the weapon used to manipulate the identified weakness to achieve the attacker's goal [54], [55].

3.4.3 Pathway

As stated in the Literature Review, the increasing use of IoT and CIs, social networks, email, mobile devices, and Web 2.0 sites has made the Internet an essential part of human life for both individuals and enterprises. The number of pathway (or target) types also increased. The types of targets include individuals, Systems and Servers, Data repositories, Critical infrastructures, and Supply chains [54].

The target is the entity or asset the attacker aims to impact, compromise, or control through the attack. Targets can range from individual users to large-scale infrastructure or nation-state systems. Targets are often selected for strategic or financial gain [56]. In some cases, attackers compromise supply chains to reach broader, higher-value targets [57].

Threat actors utilise individuals for identity theft, blackmail, or manipulation. They resort to systems and servers to gain control, carry out DDoS attacks, or engage in crypto-mining, among other activities. They target data repositories, such as databases that store Personally Identifiable Information (PII), financial data, or health records. Critical infrastructure, including energy grids, transportation systems, and water supplies, can also be targets for attacks. Additionally, in incidents such as SolarWinds® and NotPetya, attackers compromised upstream vendors to gain access to downstream clients, thereby effectively disrupting supply chains [54], [57].

The value of the asset often dictates the attacker's choice of target, the impact of compromise, or the ease of exploitation. Target selection can be opportunistic (mass scanning) or strategic (targeted APTs) [56], [57].

In short, the target is the end goal — the system, data, or person the attacker wants to compromise [54], [56], [57].

3.5 Malware Attack Vectors

The following are examples of malware attack vectors that are particularly useful; the first three are from the literature review.

3.5.1 Azolurt

Attackers developed the Azolurt malware during the COVID-19 pandemic, an information-stealing malware that targets credentials and accounts. The Azolurt is a social engineering attack vector that leverages the most common end-user fears during the reconnaissance phase, specifically the necessity for information about COVID-19 [30].

The most popular resources visited by users during the pandemic were online interactive COVID-19 maps, followed by newspapers, which ranked second among information sources. So the attackers used people's fears and created thousands of fake COVID-19 Maps and phoney news websites [30].

Threat actors delivered Azolurt via a Microsoft™ Office® document that exploited the CVE-2017-11882 Microsoft Office Equation Editor vulnerability, which then downloaded the malware.

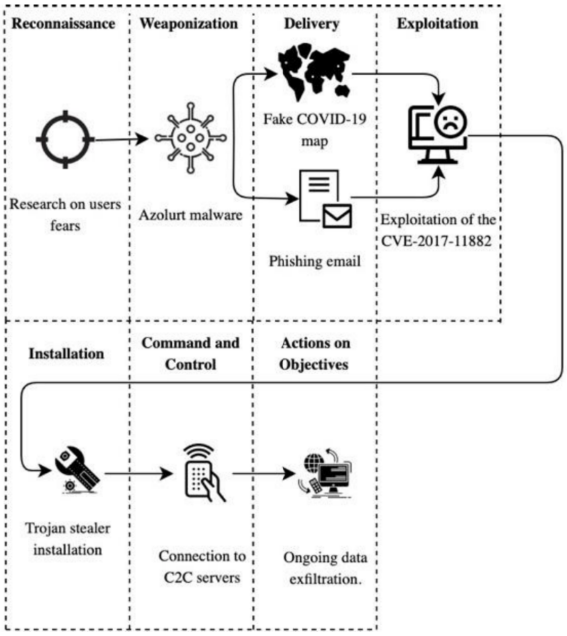
Source:'Analysis of the attack vectors used by threat actors during the pandemic' [30]

Table 3.2
Summary of Azolurt Attack.

Elements	Role	Example
Vulnerability	People's trust, fear and compassion	The user opens a Microsoft™ document
Mechanism (exploit)	Fake COVID-19 maps and news	The Azolurt malware
Target (pathway)	Stealing people's credentials and accounts	Make changes in the registry to run it at the system start

Source: 'Analysis of the attack vectors used by threat actors during the pandemic' [30]

Figure 3.3
The social engineering attack vector.



3.5.2 Watering Hole

Watering Holes are pathways that bridge mechanisms to vulnerabilities; for example, if a bad actor compromises a digital media place, where users are more vulnerable to manipulation in a familiar environment, such as a Facebook media site, they are more likely to click on links or download a funny cat video that serves as a Trojan horse malware [49].

In the watering hole attack example, the vulnerability is the user, the exploit is the Trojan horse malware, and the target is the social media site.

*Source:*FCF - Introduction to the Threat Landscape 3.0 [49]

Table 3.3
Summary of Watering Hole Attack.

Elements	Role	Example
Vulnerability	Human trust, ignorance, or poor social media defences	The user clicks a link or downloads a video
Mechanism (exploit)	Fraudulent communication (link, video)	The Trojan horse malware
Target (pathway)	The user or system attacker wants to deceive/compromise	The user Facebook

Source: FCF - Introduction to the Threat Landscape 3.0 [49]

Figure 3.4
Watering Hole on Facebook.



3.5.3 Distributed Denial-of-Service

Malware takes control of the computer, turning it into a robot. The term used to define this malware is the Bot [49]. With a bot, the bad actor controls the computer using a Command and Control (C&C) server, which sends commands to systems compromised by bot malware and then receives the stolen data. A botnet controls a collection of computers with a C&C server. As many propose, botnet threat actors use it to harvest information from infected machines to send SPAM and phishing emails, or to attack web services. An attack by a botnet against a server with the intent of making it unavailable is known as a DDoS (Figure 3.5) [49].

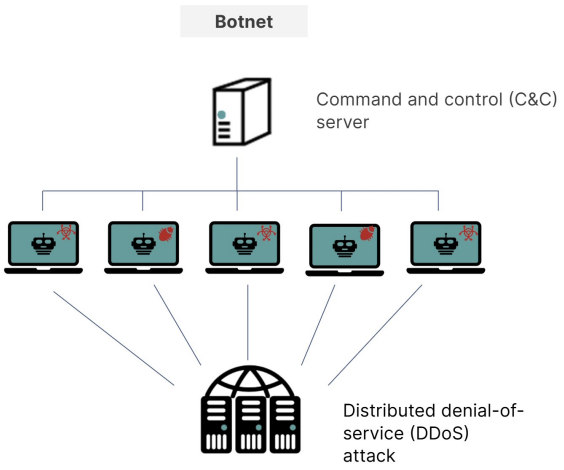
*Source:*FCF - Introduction to the Threat Landscape 3.0 [49]

Table 3.4
Summary of a Botnet Attack.

Elements	Role	Example
Vulnerability	Misconfiguration on the server	A C&C server controls a collection of computers
Mechanism (exploit)	The botnet malware	Use as many infected computers as possible on the network
Target (pathway)	They want to make the server unavailable	Internet

Source: FCF - Introduction to the Threat Landscape 3.0 [49]

Figure 3.5
Botnet attack.



3.5.4 Summary

The widespread use of the Internet has intensified the proliferation of attack vectors, resulting in a surge in cyberattacks. A consequence of an increase in cyberattacks is the impact on enterprises and people's daily lives.

The NIST and Forum of Incident Response and Security Teams, Inc. All are methods for assessing and quantifying the potential impact and likelihood of cyber threats. The scoring systems have enabled organisations to prioritise and address the most critical risks. It involves

evaluating factors related to the attack, such as its potential impact on the organisation, the likelihood of its occurrence, and the effectiveness of existing security controls [58].

Both the NIST and first.org Institutes have their frameworks for risk-scoring attacks. The CVSS v4.0 framework used in this dissertation utilises attack vectors as a metric for scoring cyberattack risk.

The section below provides a brief overview of why attack vectors are vital for scoring cyber attack risks under the CVSS v4.0 framework.

3.6 The Common Vulnerability Scoring System Framework v4.0

The purpose of the CVSS v4.0 is to provide a standardised, open framework for communicating the characteristics and severity of vulnerabilities in software, hardware, and firmware. It generates numerical scores that indicate the severity of a vulnerability, enabling organisations to assess and prioritise their vulnerability management processes. CVSS v4.0 is vendor- and platform-agnostic, providing transparency into the methodology used to derive scores. It helps organisations make informed remediation decisions by considering intrinsic, threat, environmental, and supplemental metrics related to vulnerabilities [47].

3.6.1 CVSS v4.0 Metrics

CVSS v4.0 scores vulnerabilities by evaluating them across four metric groups: Base, Threat, Environmental, and Supplemental [47].

- **Base metrics** represent intrinsic characteristics that stay constant across environments and over time. These include Exploitability Metrics (such as Attack Vectors (AVs) and Attack Complexity) and Impact Metrics (including CIA triad Confidentiality, Integrity, and Availability). The system combines Base metric values to produce a score between 0.0 and 10.0, assuming the worst-case scenario.

- **Threat Metrics** are dynamic factors, such as the availability of exploit code or active exploitation of the vulnerability.
- **Environmental Metrics** refine the score based on a user's environment, accounting for factors such as system criticality and existing mitigations.
- **Supplemental Metrics** provide context on external factors, such as safety or recovery efforts, but do not affect the final score.

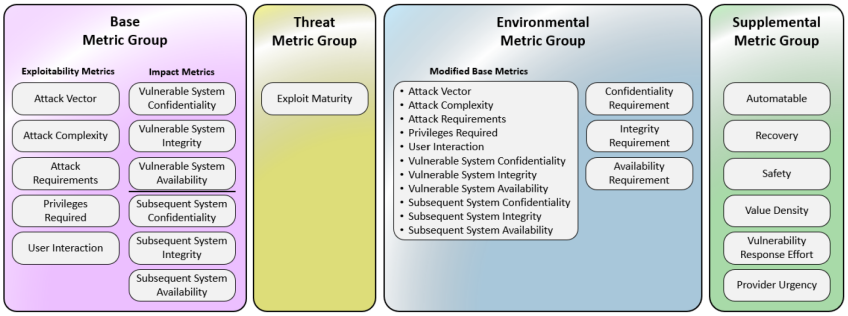
The scoring process includes:

- Representing the metric values as a concise, machine-readable vector string.
- Using predefined equations to combine Base, Threat, and Environmental metrics into a final score.
- Mapping the score to a qualitative severity rating (None, Low, Medium, High, Critical) for more straightforward interpretation.

The model assumes worst-case scenarios and treats the attacker as fully knowledgeable about the vulnerability. Organisations can enhance the Base metrics with Threat and Environmental values to reflect their specific risk levels [47].

Source: Common Vulnerability Scoring System version 4.0 Specification Document [47]

Figure 3.6
CVSS v4.0 Metric Groups.



3.6.2 CVSS v4.0 Base Metric

Base Metrics in CVSS v4.0 help assess the core attributes of a vulnerability that remain constant over time or across environments. These metrics form the basis for calculating the Base Score, which reflects the severity of a vulnerability under the worst-case scenario [47].

Exploitability Metrics measure how easily someone could exploit the flaw:

- **Attack Vector** indicates where the attack could originate (such as a network or physical access).
- **Attack Complexity** indicates the difficulty of exploiting a system.
- **Privileges Required** reflects the level of access needed.
- **User Interaction** notes whether a user must take action.

Impact Metrics evaluates what happens when an attacker succeeds:

- **Confidentiality** measures how much data is exposed.
- **Integrity** checks for data tampering.
- **Availability** covers system downtime or disruption.

These scores assume the worst-case scenario. Experts, such as analysts or vendors, use their knowledge of the vulnerability to assign values and calculate the Base Score (ranging from 0.0 to 10.0). Organisations then use Threat and Environmental Metrics to tailor the score to their specific risk landscape [47].

3.6.3 CVSS v4.0 Attack Vector Metric

The `AV` metric represents the context in which an attacker can exploit a vulnerability. It assesses how remotely an attacker can exploit the vulnerable system, assuming that vulnerabilities exploitable from greater distances pose a higher severity because they expose a larger pool of potential attackers [47].

Metric Values and Descriptions:

1. **Network (N):** The vulnerability is exploitable across a network, including the Internet. The CVSS v4.0 framework describes the network as "remotely exploitable" and assigns it the highest severity rating.

2. **Adjacent (A):** The vulnerability is exploitable within a logically adjacent topology, such as a local IP subnet, Bluetooth, or NFC.

3. **Local (L):** The vulnerability requires local access to the system, such as through a keyboard, console, or terminal emulation (e.g., SSH). It may also involve user interaction, such as opening a malicious file.

4. **Physical (P):** The attacker must physically interact with the system, such as through hardware manipulation or direct access (e.g., cold boot attacks or USB-based attacks).

Assessment Guidance:

- When deciding between a Network and an Adjacent network, an attacker determines whether to launch an attack over a vast area network or outside the logically adjacent administrative domain. They make this decision using the Network [47].

- The metric is calculated based on the attacker's proximity and the level of interaction required with the vulnerable system. Analysts should assume the attacker has advanced knowledge of the system's configuration and default defences [47].

For calculation, the value is selected based on the attacker's required proximity and the system's exposure:

- Attackers chose **Network** because it has exploitable vulnerabilities that they can remotely access over the Internet.

- **Adjacent** is selected for attacks limited to local network segments or proximity-based protocols.

- **Physical** applies to vulnerabilities that require physical manipulation of the system.

CVSS v4.0 is actively shaping how cybersecurity professionals assess and prioritise vulnerabilities in 2025. It is not just a scoring tweak; it's a strategic upgrade. CVSS v4.0 is applied today to provide more precise vulnerability scoring, improved prioritisation, support for modern systems, enhanced communication, and adoption by major players [47].

- Organisations now use CVSS v4.0 to evaluate not just the technical severity of a vulnerability but also its real-world exploitability and environmental impact. More precise vulnerability scoring helps teams focus on what truly matters in their specific context [59].
- The new Threat and Environmental metric groups allow security teams to adjust scores based on threat intelligence and their infrastructure. Improved prioritisation means a vulnerability that is critical in one environment might be less urgent in another [60].
- CVSS v4.0 better accommodates vulnerabilities in cloud, IoT, and OT systems areas, where Common Vulnerability Scoring System v3.1 (CVSS v3.1) struggled to do so [59]. For example, it introduces metrics like Attack Requirements and Subsequent System Impact, which are crucial for understanding complex, multi-layered attacks [47].
- By using standardised vector strings and detailed metrics, teams can clearly explain why a vulnerability is a high risk and what factors influenced that decision. This transparency is key for audits, compliance, and cross-team collaboration [61].
- Since its release in late 2023, CVSS v4.0 has been adopted by major cybersecurity vendors, government agencies, and vulnerability databases like the NVD. It is becoming the new normal for scoring and reporting vulnerabilities [60].

3.6.4 CVSS v4.0 Attack Vector Metric Impact on Score

The jump in score reflects the improved granularity and context-awareness of CVSS v4.0. It more accurately captures real-world risk by accounting for attack prerequisites and system-level impacts [47].

Here is how the `AV` values rank from most remote (highest risk) to most constrained (lowest risk) [62]:

- **Network (AV:N)** - Exploitable over the internet or any network. This metric yields the highest score because the attacker does not need physical or local access.
- **Adjacent (AV:A)** - Exploitable only from the same shared network segment (e.g., Bluetooth, local subnet).
- **Local (AV:L)** - Requires local access to the system (e.g., logged-in user or shell access).
- **Physical (AV:P)** - Requires physical interaction with the device (e.g., plugging in a USB). This metric gives the lowest score.

Example Impact on Score:

Let us say we have a vulnerability with the following base vector:

`CVSS v4.0:/AV:N/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:H`, where its calculation might yield a base score of 9.8 (Critical).

Now, if we change **AV:N** → **AV:L**, the score could drop to around 7.8 (High). Moreover, if we proceed to **AV:P**, it may fall to 5.9 (Medium) because the attacker now requires physical access, which is significantly harder to obtain [62].

The following Table 3.5 shows how AV has a considerable impact on the score.

*Source:*CVSS v4.0 examples [62]

Table 3.5
Example Impact on Score

Base Vector	Base Score
AV:N	9.8 (Critical)
AV:L	7.8 (High)
AV:P	5.9 (Medium)

So, **AV** is a powerful lever in CVSS v4.0 scoring. It reflects how ”reachable” a vulnerability is, and that directly affects the urgency of patching [47].

3.6.5 Real-Life CVSS 4.0 Scoring Examples

The following examples will focus only on the impact of the Attack Vector metric because Attack Vectors are the purpose of this dissertation.

A. **Example: CVE-2022-32223 – Node.js DNS Rebinding Vulnerability**

This vulnerability enables Domain Name System (DNS) rebinding attacks by allowing improper hostname validation in Node.js. An attacker could bypass same-origin policies and access internal services [62].

The CVE-2022-32223 generates the following base vector:

`CVSS v4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:L/VA:L/SC:N/SI:N/SA:N`

Breaking down:

- `AV:N` → *Attack Vector:* **Network** - the attacker can exploit the vulnerability remotely over a network.
- `AC:L` → *Attack Complexity:* **Low**.
- `AT:N` → *Attack Requirements:* **None**.
- `PR:N` → *Privileges Required:* **None**.
- `UI:N` → *User Interaction:* **None**.
- `VC:L/VI:L/VA:L` → *Low impact* on Confidentiality, Integrity and Availability.
- `SC:N/SI:N/SA:N` → *No impact* on Subsequent Systems.

From the above calculation, we get a CVSS v4.0 Score of 5.6 (Medium).

This example illustrates how CVSS v4.0 captures the nuance of a remotely exploitable, nevertheless moderately impactful, vulnerability. The `AV:N` component is critical, as it indicates that the attack can originate from anywhere on the Internet, thereby increasing the risk profile [62].

B. **Example: CVE-2024-20244 – Cisco ASA & FTD Web Services Vulnerability**

This vulnerability affects Cisco Adaptive Security Appliance (ASA) and Firepower Threat Defence (FTD) software. It allows an unauthenticated, remote attacker to cause a DoS attack by sending specially crafted HTTP packets to the web services interface [62].

The CVE-2024-20244 generates the following base vector:

`CVSS v4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:N/VI:N/VA:H/SC:N/SI:N/SA:L`

Breaking down:

- **`AV:N`** →*Attack Vector:* **Network** - the attacker can exploit the vulnerability remotely over a network.
- **`AC:L`** → *Attack Complexity:* **Low**.
- **`AT:N`** → *Attack Requirements:* **None**.
- **`PR:N`** → *Privileges Required:* **None**.
- **`UI:N`** → *User Interaction:* **None**.
- **`VC:N/VI:N/VA:H`** → *High impact* on Availability only.
- **`SC:N/SI:N/SA:L`** → *Low impact* on Subsequent Systems.

From the above calculation, we get a CVSS v4.0 Score of 7.6 (High).

This example illustrates how a network-based DoS vulnerability, requiring no user interaction or privileges, can still yield a high score due to its impact on system availability. The **`AV:N`** component is key; it indicates that an attacker can launch the attack remotely, thereby elevating the urgency to implement the corresponding patch [47].

We now examine the impact of altering the **`AV`** in the CVE-2024-20244 example and assess the resulting changes in the CVSS v4.0 score.

In the original vector, we have a Network(**`AV:N`**). Changing **`AV`** to Local(**`AV:L`**) generates the following base vector:

`CVSS v4.0/AV:L/AC:L/AT:N/PR:N/UI:N/VC:N/VI:N/VA:H/SC:N/SI:N/SA:L`

That one change, which already requires local access, significantly lowers the score to 5.8 (Medium).

3.6.6 CVSS 4.0 Attack Vector Score Equation

The **AV** metric is part of the equation used to calculate the **AV** Base Score, which reflects the intrinsic severity. The equation incorporates metrics from the Base group, including **AV**, to produce a numerical score ranging from 0.0 to 10.0 [47].

The CVSS v4.0 Base Score equation consists of two main components [62]:

1. **Exploitability Sub-Score:** This score represents the ease of exploitation of the vulnerability, encompassing metrics such as **AV**, Attack Complexity **(AC)**, Attack Requirements **(AT)**, Privileges Required **(PR)**, and User Interaction **(UI)**.

2. **Impact Sub-Score:** Represents the consequences of successful exploitation and includes metrics for Confidentiality **(VC)**, Integrity **(VI)**, and Availability **(VA)**.

Originating a general formula where [47]:

- The **Impact Sub-Score** carries more weight in the equation, as it reflects the damage caused by exploitation.

- The **Exploitability Sub-Score**, which includes **AV**, reflects the ease of exploitation.

A. Weight of Formula Components in the Basic Metrics Group Calculation

The **AV** has a significant impact on the **Exploitability Sub-Score**, as it defines the context in which exploitation occurs. The framework assigns specific weights to the values of the **AV** metric to use in the formula for the **Exploitability Sub-Score**, calculated as: **Network (N):** 0.85, **Adjacent (A):** 0.62, **Local (L):** 0.55, **Physical (P):** 0.20 [62].

The following are the other components of the equation, with assigned weights to the metric values used in the formula for the **Exploit-ability Sub-Score** [47].

- **Attack Complexity** indicates the difficulty of exploiting a system.

    - **Low (L):** 0.77
    - **High (H):** 0.44

- **Attack Requirements** reflects the prerequisite conditions of the vulnerable component that make the attack possible.
    - **None (`N`):** 1.0
    - **Present (`P`):** 0.5
- **Privileges Required** reflects the level of access needed.
    - **None (`N`):** 0.85
    - **Low (`L`):** 0.62
    - **High (`H`):** 0.27
- **User Interaction** notes whether a user must take action.
    - **None (`N`):** 0.85
    - **Passive (`P`):** 0.62
    - **Active (`A`):** 0.44

The **Impact Sub-Score** metrics are essential for understanding the potential damage a vulnerability can cause. They ensure that the CVSS v4.0 score accurately reflects the severity of the vulnerability, enabling organisations to prioritise their vulnerability management efforts effectively [47].

The **Impact Sub-Score** metrics are critical to calculating the **Base Metrics Group**, as they quantify the consequences of a successfully exploited vulnerabilities. These metrics directly reflect the severity of the impact on the vulnerable system and any subsequent systems, making them essential for determining the overall CVSS v4.0 score. Here is why they are crucial [47]:

- **`Confidentiality Impact (VC/SC)`** can be High (`H`), Low (`L`) or None (`N`). A high **`VC/SC`** score indicates a complete loss of confidentiality or access to highly sensitive information, with serious consequences. A low score indicates a partial loss of confidentiality, with limited access to restricted information and minor consequences. A score of zero (`none`) indicates no loss of confidentiality [47].

- **`Integrity Impact (VI/SI)`** can be High (**`H`**), Low (**`L`**), or None (**`N`**). A high score of **`VI/SI`** indicates a total loss of confidentiality or unauthorised modification of data or files with serious consequences. A low score indicates a partial modification of data, limited in scope or consequence. A score of zero (`none`) suggests no loss of integrity [47].

- A **`Availability Impact (VA/SA)`** can be High (**`H`**), Low (**`L`**), or None (**`N`**). A high score of **`VA/SA`** indicates a total loss of availability, resulting in complete denial of access to resources or services. A low score suggests reduced performance or intermittent availability, but no serious consequences. A score of zero (`none`) means no loss of availability [47].

The Subsequent System Impact, for impacts outside the vulnerable system (e.g., downstream systems), is categorised into three key areas: Confidentiality Impact (**`CI`**), Integrity Impact (**`II`**), and Availability Impact (**`AI`**). The mentioned areas have value metrics: High (**`H`**), Low (**`L`**), and None (**`N`**) [47].

The CVSS v4.0 framework uses the following values to calculate the Impact Sub-Score, which reflects the severity of the consequences of a successful exploit. The Impact Sub-Score metrics in CVSS v4.0 are [47]:

- **Vulnerable System Confidentiality (`VC`):**
  - **None (`N`):** 0.0
  - **Low (`L`):** 0.22
  - **High (`H`):** 0.56
- **Vulnerable System Integrity (`VI`):**
  - **None (`N`):** 0.0
  - **Low (`L`):** 0.22
  - **High (`H`):** 0.56
- **Vulnerable System Availability (`VA`):**
  - **None (`N`):** 0.0

  - **Low (L):** 0.22
  - **High (H):** 0.56

- **Subsequent System Confidentiality (SC):**

  - **None (N):** 0.0
  - **Low (L):** 0.22
  - **High (H):** 0.56

- **Subsequent System Integrity (SI):**

  - **None (N):** 0.0
  - **Low (L):** 0.22
  - **High (H):** 0.56

- **Subsequent System Availability (SA):**

  - **None (N):** 0.0
  - **Low (L):** 0.22
  - **High (H):** 0.56

B. Base Score Overview

The above information is crucial for the CVSS v4.0 base metric calculations for various CVEs. It is now possible to calculate the base score for several real-world examples [62].

Analysts calculate the Base Score from the Impact Sub-Score (ISS) (Vulnerable & Subsequent Systems) and ESS (ease of exploitation) [62].

The ISS formula is:

$$ISS = 6.42 \times ISC_{base}$$

where

$$ISC_{base} = 1 - [(1 - C) \times (1 - I) \times (1 - A)]$$

The C, I, and A represent `Confidentiality Impact (C)`, `Integrity Impact (I)`, and `Availability Impact (A)`, respectively. Each of these letters can have one of three values: High (0.56), Low (0.22), or None (0.0) [62].

That means: if all $C, I, A = 0 \rightarrow ISC_{base} = 0$, if one or more have impact $\rightarrow ISC_{base}$ increases (non-linear growth) [62].

The Subsequent System Impact Metrics (`SC`, `SI`, `SA`) capture the impact on subsequent systems that might be indirectly affected. CVSS v4.0 allows for scaling the original ISC_{base} by: $ISS = 6.45 \times ISC_{base} + 1$ (if the Scope is Changed). However, in CVSS v4.0, the "Scope Changed" adjustment no longer applies as it did in CVSS v3.x [62].

In CVSS v4.0, analysts calculate the Exploitability Sub-Score to determine how easily attackers can exploit a vulnerable system. This sub-score is derived exclusively from four metrics: Attack Vector **(AV)**, Attack Complexity **(AC)**, Privileges Required **(PR)**, and User Interaction **(UI)** (Table 3.6). CVSS v4.0 defines the formula as [62]:

$$ESS = 8.22 \times AV \times AC \times PR \times UI$$

*Source:*CVSS v4.0 examples [62]

Table 3.6
CVSS v4.0 Exploitability Sub-Score

Metrics	Values (examples)
Attack Vector (AV)	Network=0.85, Adjacent=0.62, Local=0.55, Physical=0.2
Attack Complexity (AC)	Low=0.77, High=0.44
Privileges Required (PR)	None=0.85, Low=0.62, High=0.27 (adjusted if Scope is changed)
User Interaction (UI)	None=0.85, Required=0.62

This formulation mirrors the structure found in CVSS v3. x and remains unchanged in CVSS v4.0. Notably, the metric Attack Requirements **(AT)**, introduced by the CVSS v4.0 framework to capture relevant conditions external to the target system (such as specific configurations or preparatory steps required by an attacker), does not affect the Base Score or the Exploitability Sub-Score. Instead, **AT** contributes only to the calculation of the Threat and Environmental Scores. This distinction preserves the Base Score as a measure of the inherent technical severity, independent of contextual prerequisites [62].

The following formula calculates the Base Score:

$$\text{BaseScore} = \begin{cases} 0, if ISC_{base} = 0 \\ Roundup(nin(ISS + ESS), 10), otherwise \end{cases}$$

Where

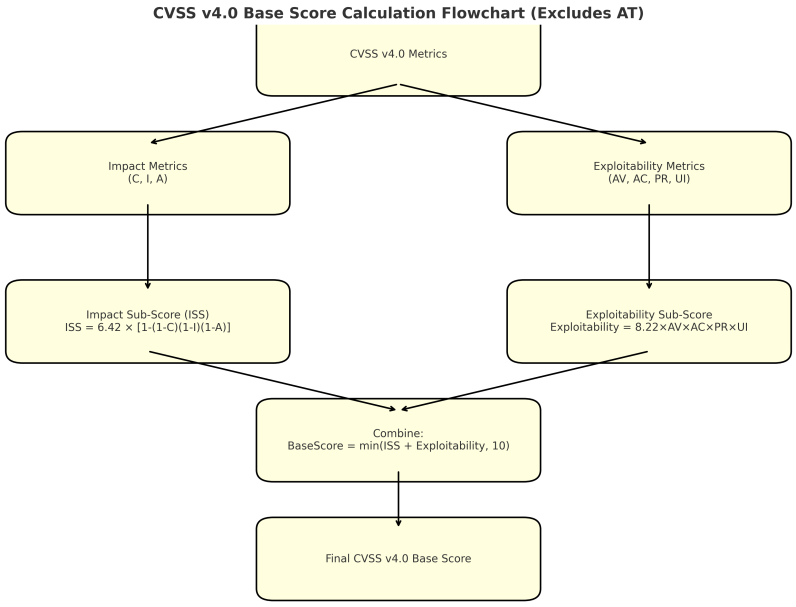
$$BaseScore = Roundup(nin(ISS + ESS)$$

The '*RoundUp*' means rounding up to 1 decimal (e.g., 4.02 → 4.1).

*Source:*CVSS v4.0 examples [62]

Figure 3.7

CVSS v4.0 Base Score Calculation Flowchart.



The Figure 3.7 illustrates the calculation process for the CVSS v4.0 Base Score, showing the separation between Impact Metrics and Exploitability Metrics. The exclusion of **AT** from this calculation reinforces the Base Score's focus on intrinsic vulnerability properties [47].

The Table 3.8 summarises two vectors and their scores. Firstly, we present a step-by-step calculation of the score for the second vector string in the table. The Table 3.7 shows the vector string metrics to calculate the Base Score.

Vector String for CVE-2023-3089: Compliance Problem in Red Hat OpenShift Container Platform

```
CVSS v4.0/AV:N/AC:L/AT:P/PR:N/UI:N/VC:H/VI:L/VA:L/SC:N/SI:N/SA:N
```

*Source:*CVSS v4.0 examples [62]

Table 3.7
Vector String Metrics.

Metrics	Values
Attack Vector (AV):N	0.85
Attack Complexity (AC):L	0.77
Attack Requirements (AT):P	Preparation required; irrelevant for Base Score.
Privileges Required (PR):N	0.85
User Interaction (UI:N)	0.85
Vulnerable System Impacts	
Confidentiality (VC:H)	0.56
Integrity (VI:L)	0.22
Availability (VA:L)	0.22
Subsequent System Impacts (SC, SI, SA)	None for all. These are not included in the Base Score.

Impact Sub-Score (ISS)

$$ISC_{base} = 1 - (1 - 0.56) \cdot (1 - 0.22) \cdot (1 - 0.22)$$

$$ISC_{base} = 1 - (0.44) \cdot (0.78) \cdot (0.78)$$

$$ISC_{base} = 1 - 0.267 = 0.733$$

$$ISS = 6.42 \cdot 0.733 = 4.71$$

Exploitability Sub-Score (ESS)

$$ESS = 8.22 \cdot AV \cdot AC \cdot PR \cdot UI$$

$$ESS = 8.22 \cdot 0.85 \cdot 0.77 \cdot 0.85 \cdot 0.85$$

$$ESS \approx 8.22 \cdot 0.475 = 3.90$$

Base Score

$$BaseScore = min(ISS + Exploitability, 10.0)$$

$$BaseScore = min(4.71 + 3.90, 10.0)$$

$$BaseScore = 8.61 \rightarrow Roundedup = 8.7High$$

*Source:*CVSS v4.0 examples [62]

Table 3.8

CVSS v4.0 Vector Strings with Corresponding Base Scores and Severity Ratings

Vector String	Impact Sub-Score	Exploitability Sub-Score	Base Score	Severity
CVE-2024-20244 `CVSS v4.0/AV:L/AC:L/AT:N/PR:N/UI:N/VC:N/VI:N/VA:H/SC:N/SI:N/SA:L`	3.60	3.905	**7.6**	High
CVE-2023-3089 `CVSS v4.0/AV:N/AC:L/AT:P/PR:N/UI:N/VC:H/VI:L/VA:L/SC:N/SI:N/SA:N`	4.71	3.90	**8.7**	High

The first vector represents a vulnerability requiring local access (**`AV:L`**), with low attack complexity (**`AC:L`**) and no requirement for privileges or user interaction (**`PR:N, UI:N`**). The only impact metric contributing to the Impact Sub-Score is high availability impact (**`VA:H`**), resulting in an ISC_{base} of 0.56 and an Impact Sub-Score of 3.60. The Exploitability Sub-Score, determined by the product of metric values for **`AV, AC, PR`**, and **`UI`**, totals 3.905. When combined, the resulting Base Score of 7.6 categorises the vulnerability as high severity under CVSS v4.0. This score highlights the limited risk posed by the local attack vector, despite its significant impact on availability [62].

The second vector describes a remotely exploitable vulnerability (**`AV:N`**) with low attack complexity (**`AC:L`**) and no need for privileges or user interaction (**`PR:N, UI:N`**). The impact metrics reflect high confidentiality impact (**`VC:H`**), low integrity (**`VI:L`**), and low availability (**`VA:L`**). These produce an ISC_{base} of 0.733 and an Impact Sub-Score of 4.71. The Exploitability Sub-Score is higher than in the previous vector, at 3.90, reflecting the remote attack vector and the ease of exploitation. The combined Base Score of 8.7 indicates a high-severity vulnerability. This result highlights the increased risk posed by remote network-based attacks, which can compromise confidentiality and system integrity [62].

This comparative analysis demonstrates how differences in attack vector and system impact metrics significantly influence the resulting CVSS v4.0 Base Score. In particular, remote accessibility and high confidentiality impact elevate the severity rating in the second scenario.

3.7 Summary

This chapter analysed attack vectors in detail, defining their structure, components, and operational characteristics within the context of web security. It examined how attackers exploit vulnerabilities through specific pathways, including malware distribution, social engineering, and network-based intrusion techniques.

The chapter also introduced the elements that constitute an attack vector, such as the entry point, payload, delivery mechanism, and impact. This study focuses on malware-based vectors and their role in modern cyber threats. Additionally, this study discusses the CVSS v4.0 framework as a standardised method for evaluating vulnerability severity and exploitability.

By combining theoretical definitions with practical examples, this chapter provides a structured understanding of how attack vectors operate and how analysts can assess them quantitatively. These insights informed the design of realistic attack scenarios implemented in the CyberForge platform.

In summary, the chapter established a comprehensive conceptual framework for analysing web attack vectors, which underpins the methodological approach and system design described in the subsequent chapters.

Chapter 4

Project - CyberForge as an Educational Platform

4.1 Introduction

As described in the Chapter 3, Attack Vectors are crucial to an attacker's success or failure in exploiting vulnerabilities. The use of the Internet or any network enables attackers to develop specific attack methods and categorise them into subcategories. CVSS v4.0 Base Score demonstrates that the possibility of accessing systems from the Internet increases the risk.

Based on the Literature Review, the research theoretical component of this dissertation, including the selected sections from each author that support our dissertation, analyses the main challenges in cybersecurity education that led to the creation of the CyberForge platform. Many institutions still depend on static materials or closed exercises that do not replicate real-world attack dynamics. Students often struggle to apply theoretical knowledge in practical contexts. The intent was to create an interactive educational platform that enhances both technical proficiency and cybersecurity awareness. The described limitations are addressed by simulating realistic web-based attack environments, enabling experiential learning in which learners interact with simulated vulnerabilities, test defences, and observe system responses. This approach aligns with constructivist learning principles, holding that knowledge is constructed through direct experience and reflection [3].

The CyberForge concept combines pedagogical and symbolic inspiration. The name reflects both the cybersecurity domain central to this MSc project and the author's long-standing interest in medieval weapon craftsmanship. Just as a swordsmith forges a sword through heat, repetition, and refinement, similarly, CyberForge strengthens learners' understanding through repeated practice with real-world web vulnerabilities. Each lab serves as a stage in this process of forging, gradually shaping the student's technical and analytical skills.

CyberForge platform integrates offensive and defensive cybersecurity principles through a controlled, gamified framework [3], [4], [63], [64] that allows participants to explore vulnerabilities, exploit mechanisms, and test mitigation strategies in an ethically supervised context.

From a pedagogical standpoint, CyberForge embodies the principles of experiential learning, in which students acquire knowledge through direct engagement with real-world scenarios rather than solely through theoretical exposition. By replicating authentic web vulnerabilities aligned with the OWASP Top Ten [5]—such as SQL Injection, XSS, and Insecure Direct Object References (IDOR)—the platform enables students to understand how security flaws arise from coding practices, configuration errors, or insufficient validation. This practical exposure fosters deeper conceptual understanding, bridging the gap between abstract cybersecurity theory and applied practice.

Furthermore, CyberForge functions as an *awareness-and-behavioural-reinforcement mechanism* [65], [66], [67], [68], [69]. It provides a safe, isolated laboratory environment where developers, system administrators, and students can observe the full attack lifecycle: reconnaissance, exploitation, privilege escalation, and mitigation. Through guided challenges and immediate feedback, users not only learn to identify vulnerabilities but also internalise secure development principles, thereby improving their defensive mindset.

CyberForge adopts a constructivist learning paradigm in which learners actively build their understanding of security concepts through experimentation, hands-on exploration, and reflective analysis [70], [71], [72]. This approach follows experiential learning principles by encouraging students to learn through direct action, deliberate trial-and-error, and the interpretation of outcomes [73], [74], [75], [76]. Research in cybersecurity education shows that cyber ranges support this process by providing structured, authentic environments that

promote skill development and progressive mastery [3], [77], [78]. CyberForge reinforces these pedagogical foundations by integrating interactive dashboards, scoring systems, and learning analytics that prompt continuous engagement and enable students to evaluate their performance in real time. These mechanisms, supported by evidence on gamification and learning analytics [4], [63], [64], transform cybersecurity training into an iterative, measurable, and student-centred learning process.

CyberForge enhances cybersecurity awareness by demonstrating the tangible impact of insecure coding on real systems. The platform shows that awareness functions most effectively as an experiential process rather than a purely informational one, as users develop a deeper understanding of threats when they witness their consequences in a controlled, sandboxed environment [65], [66], [67]. This experiential approach supports preventive behaviour: once learners observe how attackers exploit weaknesses, they are more likely to adopt secure-coding habits and system-hardening practices in professional settings [68], [69].

CyberForge also contributes to broader organisational and societal objectives by strengthening cyber resilience through education. Research shows that simulation-based learning, cyber ranges, and hands-on exercises significantly enhance security readiness and behavioural change [78], [79]. By integrating simulation, gamification, and continuous assessment, CyberForge transforms cybersecurity training from a passive theoretical subject into an active, immersive, and reflective process. This model cultivates a sustained culture of security awareness that extends beyond the classroom and influences professional development, organisational practices, and institutional defence strategies [3], [4], [63], [64].

With the conceptual design and requirements established, the next step involves translating the proposed solution into a concrete system. Chapter 5 therefore describes the technical implementation of CyberForge, detailing its architecture, modules, and deployment environment.

4.2 Methodology

This section outlines the methodological approach used to design, develop, and evaluate the *CyberForge*. This educational platform teaches secure web development and attack

analysis through interactive, web-based scenarios. The methodology follows a structured, iterative workflow that integrates requirements engineering, secure system design, virtualisation architecture, and empirical validation through practical cybersecurity scenarios.

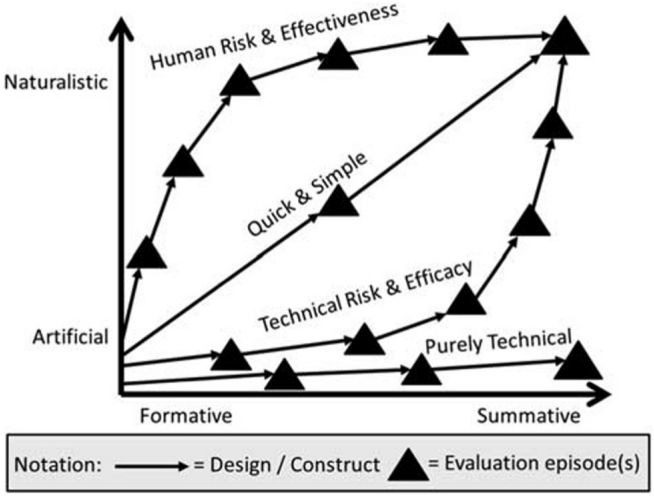
The methodology employs a mixed-methods approach, which captures both quantitative and qualitative insights. Quantitative data include task completion times, interaction frequencies, and success rates in exploiting or mitigating vulnerabilities. Qualitative observations focus on user feedback, perceived engagement, and learning progression [80]. This combination ensures a comprehensive evaluation of both the platform's technical reliability and its pedagogical impact.

This approach requires the system to combine theoretical foundations in web security with practical insights from hands-on attack simulation. The platform thus emerges from cycles of problem analysis, design, implementation, and evaluation, each cycle refining the artefact toward usability, security, and educational impact.

The methodology follows five iterative stages: problem identification, objective definition, design and development, demonstration, and evaluation [81].

Source: 'FEDS: A Framework for Evaluation in Design Science Research' [81]

Figure 4.1
Framework for Evaluation in Design Science Research (FEDS).

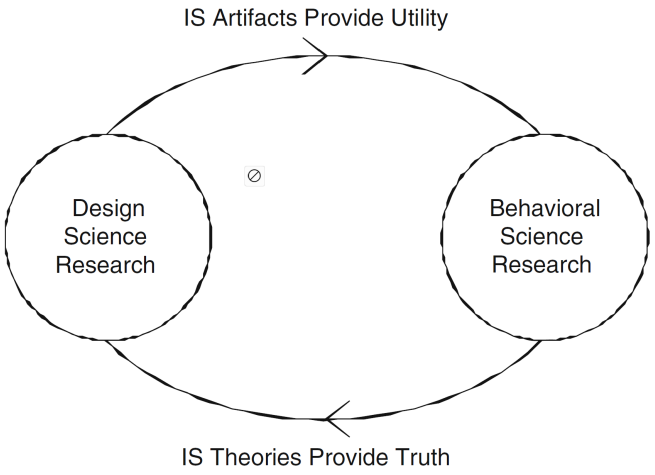


4.2.1 Research Approach

The development of CyberForge adopted a design-science methodology, an iterative cycle inspired by Agile principles [82], [83], aiming to produce an artefact that solves a concrete pedagogical and technical problem. The design-science paradigm aligns with this project because it emphasises the creation and refinement of artefacts that solve real-world issues [84].

Source: Design Research in Information Systems: Theory and Practice [84]

Figure 4.2
Complementary nature of design science research and behavioural science research.



In this project, the artefact is CyberForge. This platform functions as both a learning environment and a security training laboratory, enabling students to practise realistic web attacks safely and reproducibly. Each iteration introduced new features or refinements based on user testing and trainer observations. This cycle ensured continuous improvement and adaptability to user experience [82]. The evaluation phase involved real students enrolled in cybersecurity courses who engaged with the platform under controlled laboratory conditions.

4.2.2 Scenario-Driven Development

To ensure alignment with real cybersecurity practices, the development process uses a scenario-driven design approach. Each vulnerability scenario—such as SQL Injection, IDOR,

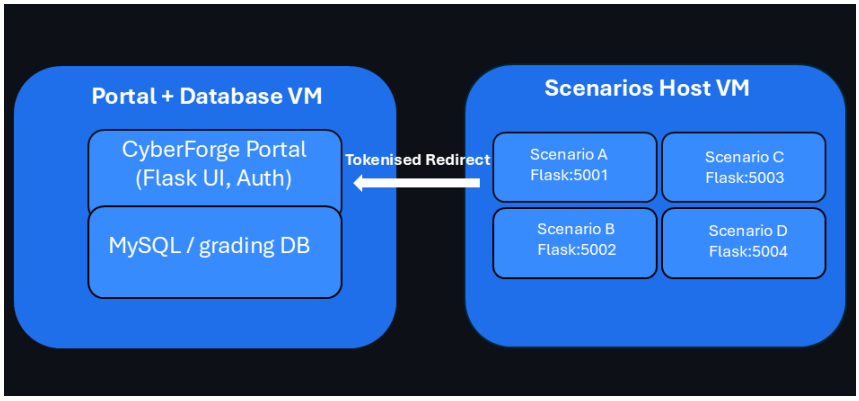
XSS or Cross-Site Request Forgery (CSRF)—defines specific learning objectives, attack paths, expected outcomes, and grading metrics. These scenarios guide the functional requirements for logging, session tracking, user isolation, and attack-replication guarantees, ensuring that every system component directly supports the learning process.

4.2.3 Virtualised Experimental Environment

This project adopts virtualisation as its primary experimental technique, enabling precise and ethical evaluation of web application vulnerabilities in a controlled environment. Virtual machines provide firm isolation boundaries that prevent unintended interactions with external systems while supporting realistic replication of application behaviour [85], [86]. Containers and lightweight Flask instances further enhance reproducibility and consistency across repeated executions [87], [88].

In alignment with established principles of cybersecurity experimentation, the platform ensures controlled execution, repeatable scenario deployment, and fine-grained forensic data collection for analytical and pedagogical purposes [89], [90]. To operationalise these requirements, the system employs a two-VM architecture: one VM hosts the CyberForge portal and its MySQL grading database, while the other hosts all vulnerable Flask scenarios on isolated ports. This separation preserves security and isolation, simplifies orchestration, and aligns with recommended practices for safe and legally compliant cyber-range design [78], [91].

Figure 4.3
Two-VM architecture.



4.2.4 Evaluation Methodology

CyberForge employs a structured, multi-layered evaluation approach that draws on established practices in software evaluation, cybersecurity pedagogy, and cyber-range engineering.

Evaluation occurs through three complementary methods:

1. **Functional validation** – confirming that each scenario launches correctly, isolates users, enforces time limits, and produces accurate logs [92], [93].
2. **Pedagogical validation** – assessing whether students can complete scenarios as designed and whether logs and scoring correctly reflect their actions [3], [77], [94].
3. **Security validation** – ensuring isolation, cleanup, correct session handling, and prevention of unauthorised access across user environments [78], [95], [96].

This multi-layered evaluation ensures that CyberForge meets both academic and operational standards for cybersecurity training.

4.3 Requirement Analysis

This section analyses the functional, architectural, and operational requirements that underpin the CyberForge platform's design. It examines the threat model, core interaction workflow, and system topology to establish a clear foundation for implementation.

4.3.1 Threat Model: Web Application as Victim and Browser as Attacker

CyberForge models the web application as the victim while treating the browser as the primary attack vector. This perspective aligns with real-world security scenarios, where adversaries frequently exploit client-side inputs, Document Object Model (DOM) interactions, and HTTP requests to compromise application logic. By framing the browser as an active

adversarial agent, the system highlights the trust boundaries modern applications must enforce and demonstrates how seemingly benign user interactions can initiate harmful operations. This approach ensures that learners engage with scenarios that accurately reflect contemporary attack methodologies and the security implications of poorly validated or untrusted client behaviour.

4.3.2 Operational Workflow: Scenario Launch Sequence

Operationally, the platform defines scenario launch as an atomic, auditable workflow that begins with an explicit user action, typically a click on a "Launch" button. Once the user initiates this action, the system executes a deterministic sequence:

1. It verifies user authentication and authorisation.
2. It sends a scenario allocation request to the orchestration layer.
3. It generates and records a unique session token and ephemeral identifiers.
4. It initiates resource provisioning, including VM or container allocation, network namespace creation, and service startup.
5. It returns the scenario endpoint and access metadata (IP address, port, credentials, and expiry time) to the user.
6. It initiates comprehensive monitoring and logging for grading and audit purposes.

4.4 Existing Solutions

Several established platforms and pedagogical tools exist within the domain of web security education, each contributing distinct approaches to vulnerability demonstration, hands-on practice, and attack simulation. These systems provide the foundational context against which CyberForge can be situated and evaluated. Although they differ in architecture, scope, and

intended audience, they all share the common objective of enhancing cybersecurity competencies through experiential learning.

4.4.1 OWASP Juice Shop

One of the most widely recognised platforms is the OWASP Juice Shop, an intentionally vulnerable web application designed to encompass the full spectrum of the OWASP Top Ten vulnerabilities. Juice Shop is notable for its realism, breadth, and gamified structure, offering challenges of varying difficulty and automatic scoring [97]. It functions as both a training environment for security professionals and a pedagogical tool for developers seeking to understand secure coding practices. However, it remains a single-application model, meaning that each learner interacts with the same underlying platform instance unless the system uses additional containerisation or orchestration.

4.4.2 Damn Vulnerable Web Application

Another prominent tool is Damn Vulnerable Web Application (DVWA), a PHP/MySQL-based environment that presents vulnerabilities at multiple difficulty levels. Its value lies in its modularity and simplicity: clearly separated individual weaknesses such as SQL Injection, XSS, or CSRF, enabling targeted instruction and controlled experimentation [98]. Despite its educational utility, DVWA lacks the realism of modern production architectures and requires manual resetting to a safe baseline, making it unsuitable for large-scale classroom deployment.

4.4.3 WebGoat

WebGoat, developed by OWASP, adopts a more tutorial-driven methodology. It provides step-by-step lessons on understanding and exploiting web vulnerabilities, including interactive exercises and theoretical explanations [99]. WebGoat is pedagogically strong but somewhat linear; its lesson-based structure makes it less suitable for open-ended attack simulations or competitive capture-the-flag (CTF) style environments.

4.4.4 Others

Beyond standalone vulnerable applications, several platforms offer broader virtualised ecosystems. Hack The Box, TryHackMe, and PortSwigger's Web Security Academy provide cloud-hosted, on-demand challenges designed for professional skill development [100], [101], [102]. These services utilise dynamic provisioning, isolated virtual machines, and structured learning paths, enabling participants to practice attacks in a controlled yet highly realistic environment. While powerful, these platforms are externally hosted, not locally deployable, and often run on proprietary architectures, limiting their adaptability for academic institutions that require custom, curriculum-aligned content or offline operation.

Capture-the-flag frameworks such as CTFd and RootTheBox support the creation of custom challenges and competition environments [103], [104]. Research has shown that CTF competitions are effective environments for developing practical cybersecurity skills [80]. These platforms can manage scoring, user accounts, and challenge distribution, but lack built-in functionality for automatically provisioning isolated environments or spinning up independent vulnerable systems per user. As a result, institutions frequently combine CTF platforms with external orchestration layers to achieve the desired level of isolation and reproducibility.

Collectively, these tools demonstrate a growing focus on immersive, hands-on cybersecurity learning. However, they also reveal several limitations that motivate the development of new solutions such as CyberForge: the absence of a per-user isolated environment in many platforms, limited integration with institutional grading workflows, insufficient realism in some pedagogical tools, and the lack of unified mechanisms for scenario lifecycle management, logging, and automatic assessment. CyberForge addresses these gaps by combining scenario provisioning, attack simulation, user-specific environments, and educational analytics into a coherent, institution-controlled platform.

4.4.5 Comparison

When compared with existing educational platforms, CyberForge distinguishes itself through its integrated orchestration model, per-user scenario isolation, and alignment with institutional assessment requirements—unlike OWASP Juice Shop, DVWA, and WebGoat—which primarily

operate as single, shared vulnerable applications—CyberForge provisions dedicated virtual environments or containers for each learner, eliminating cross-user interference and enabling realistic exploitation without risking impact on other participants. Cloud-based services, not tightly coupled to academic workflows, grading systems, or offline laboratory infrastructure, such as Hack The Box, TryHackMe, and the Web Security Academy, provide advanced, externally hosted, closed-source vulnerability labs. Similarly, CTF platforms like CTFd provide challenge management and scoring but lack native capabilities for automated VM lifecycle management, isolated networking, or scenario-specific logging.

CyberForge bridges these gaps by combining the pedagogical strengths of vulnerable applications with the operational advantages of dynamic provisioning: each scenario is launched on demand through a unified portal, tied to a specific student, logged with granular detail, and automatically assessed. This approach not only enhances educational fidelity but also ensures reproducibility, scalability, and institutional control over content, making CyberForge a more complete and academically adaptable platform than existing fragmented solutions.

4.5 Proposed Solution

4.5.1 Introduction

This section proposes *CyberForge*, the primary artefact developed in this dissertation. CyberForge provides a controlled, measurable platform for practising web-application security that bridges the recurring gap between conceptual instruction and practical exploitation and remediation of vulnerabilities. The platform combines virtualisation, scenario-driven design, token-governed access control, and structured telemetry collection to support safe experimentation, reproducibility, and evidence-based assessment.

4.5.2 Design Objectives and Success Criteria

CyberForge follows six design objectives that operationalise the requirements identified earlier in this dissertation:

1. **Realistic vulnerability simulation:** represent web vulnerabilities in executable applications (e.g., SQL injection, XSS, IDOR) that preserve realistic request flows and outcomes [5], [55].
2. **Isolation and containment:** prevent learner actions from impacting other learners, the control plane, or external systems, while retaining pedagogically valuable feedback [91].
3. **Reproducibility:** allow repeatable execution of the same scenario under comparable conditions to support learning and evaluation [89].
4. **Measurable assessment:** record sufficient evidence to support automated scoring and instructor review [94].
5. **Operational efficiency:** minimise provisioning overhead and simplify maintenance for classroom-scale use [96].
6. **Ethical and legal safeguards:** enforce boundaries consistent with safe cyber experimentation and cyber-range guidance [78], [89].

Success criteria include deterministic scenario launch, enforced time limits, reliable logging, robust token validation, and demonstrated alignment between scoring outputs and learner actions.

4.5.3 System Architecture

A. Two-VM Separation Model

CyberForge adopts a two-VM architecture to separate the control plane from the execution plane:

1. **VM1 (Portal and Database Layer):** hosts the CyberForge portal, authentication/authorisation logic, orchestration services, and the MySQL grading database.
2. **VM2 (Scenario Host Layer):** hosts the vulnerable Flask scenarios, each bound to a dedicated TCP port (e.g., 5001, 5002, 5003).

The CyberForge architecture separates the portal and scenario execution environments using two dedicated virtual machines. This separation reduces the attack surface of administrative services by preventing vulnerable applications from executing on the portal and database host, thereby protecting critical management components [105]. Practitioners commonly apply similar architectural principles in cyber-range infrastructures, where control and orchestration components are isolated from experimental environments to ensure safe and reproducible experimentation [89], [91]. In addition, separating management and scenario execution services simplifies monitoring and maintenance by centralising orchestration and persistent data within the portal VM, while delegating vulnerability execution to dedicated scenario-host VM [96].

B. Port-Based Scenario Hosting

CyberForge runs each scenario as an independent Flask application bound to a unique port on VM2. Port separation provides a clear and operationally lightweight mechanism for isolating scenario endpoints [106]. The platform complements this approach with token-based access control and firewall or reverse proxy policies to prevent unauthorised direct access to scenario ports [107].

4.5.4 Scenario Lifecycle and Orchestration

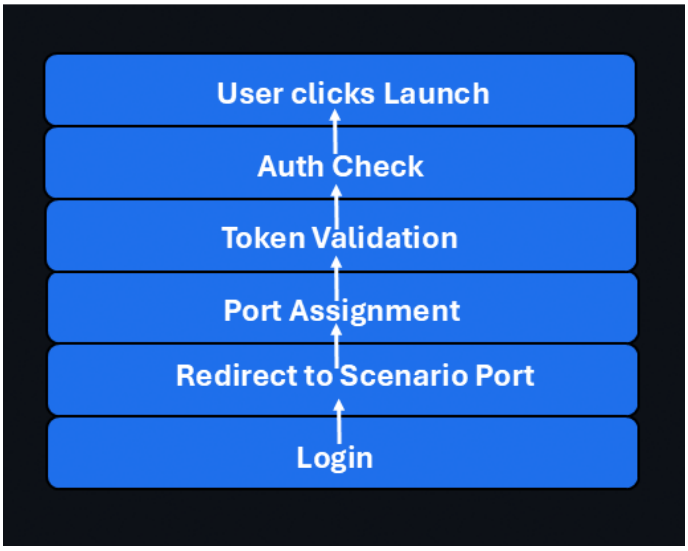
A. Launch Workflow

When a learner initiates a scenario, the portal executes a deterministic workflow:

1. Authenticate and authorise the user.
2. Resolve the requested scenario identifier to the configured port on VM2.
3. Generate a cryptographically strong, time-limited session token.
4. Persist allocation metadata and session state in the grading database.
5. Redirect the learner to the scenario endpoint: `http://scenario-host:port/?token=<token>`

This workflow, as shown in Figure 4.4, avoids the per-session VM provisioning overhead while preserving a clear execution boundary between portal functions and vulnerable services.

Figure 4.4
Launch Worflow.



B. Session Tokens and Time-to-Live

CyberForge encodes the following attributes into each session token:

- **User identity** (user ID);
- **Scenario identity** (scenario ID);
- **Token** (a unique SHA-256 hash generated when the lab starts);
- **Expiry time** (TTL);
- **Integrity protection** (IP Address, signature or MAC).

VM2 validates each token on incoming requests (e.g., via a Flask `before_request` hook) [108]. The system denies requests with missing, invalid, or expired tokens and logs the event for audit. This design supports both access control and evaluation integrity by binding evidence to a specific user and scenario instance.

4.5.5 Isolation, Containment, and Threat Model

A. Threat Model Assumptions

CyberForge assumes that learners act as semi-trusted users who may attempt to: (i) access scenarios for which they are not authorised; (ii) bypass time limits. (iii) tamper with scoring mechanisms; or (iv) interfere with other learners. The platform, therefore, treats the browser as the primary attack vector and enforces strict trust boundaries between VM1 and VM2, as well as within each VM [109], [110].

B. Controls

CyberForge enforces containment through layered controls:

- **Plane separation**: VM1 hosts control and persistence; VM2 hosts vulnerable services.
- **Port policy**: VM2 exposes only the required scenario ports; network policy restricts access paths.
- **Token gating**: scenarios accept only authorised, non-expired tokens.
- **TTL enforcement**: time limits apply to access and scoring windows.
- **Audit logging**: the system logs allocations, denials, and exploit evidence.

This model follows cyber experimentation guidance that emphasises controlled execution, monitoring, and isolation [78], [89].

4.5.6 Pedagogical Model and Learning Experience

A. Constructivist and Experiential Learning Alignment

CyberForge structures learning as an active process in which learners build understanding through experimentation and reflection [70], [71], [73]. Each scenario guides learners through observable cause-and-effect relationships between insecure design choices and exploitation

outcomes. This approach supports safe failure in a sandboxed environment, enabling iterative refinement of hypotheses and techniques.

B. Secure-Coding Awareness as Experiential Practice

CyberForge strengthens secure-coding awareness by making the impact of insecure coding observable. Learners observe how specific implementation decisions enable attack paths and connect exploit evidence to defensive countermeasures. Prior work indicates that realistic exposure to security pitfalls and feedback improves developer security behaviour [68].

4.5.7 Gamification, Dashboards, and Assessment

A. Gamification and Motivation

CyberForge integrates structured scoring and progression elements (e.g., points, completion states, and optional leader-boards) to encourage sustained engagement. Gamification research supports the use of game elements to promote motivation and persistence in learning tasks [63], [64].

B. Dashboards and Learning Analytics

The portal presents dashboards that summarise learner progress and performance metrics. Learning analytics dashboards can promote self-assessment and reflection by making progress and outcomes visible [4]. In CyberForge, analytics operate at two levels: learner-facing indicators (completion, time, score) and instructor-facing evidence (payloads, timestamps, success criteria).

C. Evidence Model for Scoring

CyberForge derives scores from auditable evidence, including scenario-specific success markers, HTTP interaction logs and payloads, time-to-completion metrics, and token/metadata linking actions to users. This evidence-based approach reduces subjectivity and supports transparent grading aligned with cybersecurity lab pedagogy [77].

4.5.8 Implementation Overview

A. Portal and Database (VM1)

VM1 hosts the Flask portal, the orchestration endpoint (e.g., `/start_lab`), and the MySQL database. The portal stores scenario metadata (ID-to-port mapping), user enrolments, session states, scoring rules, and audit logs.

B. Scenario Host (VM2)

VM2 runs multiple Flask processes, one per scenario. Each process implements token validation middleware, scenario-specific vulnerable logic, per-scenario logging, and explicit TTL checks. This structure enables rapid access to multiple scenarios without the overhead of provisioning a VM per learner session.

4.5.9 Evaluation Strategy

CyberForge evaluates platform quality through three complementary methods. Functional validation confirms deterministic scenario launch, correct port assignment, TTL enforcement, and accurate logging [92]. Pedagogical validation examines whether learners can complete scenarios as designed and whether logs and scoring reflect learner actions [77], [111]. Security validation tests token enforcement, isolation boundaries, clean-up routines, and prevention of unauthorised access across user environments [78], [89].

4.5.10 Design Trade-offs and Limitations

CyberForge trades hypervisor-level per-user isolation for operational efficiency by hosting scenarios as multiple Flask services on a dedicated scenario VM. This decision reduces provisioning time and resource consumption, which supports classroom concurrency. However, it increases reliance on correct application-layer isolation, rigorous token validation, and restrictive network policies. The design therefore applies a defence-in-depth strategy by combining VM-level separation (VM1 vs VM2), port policy, token gating, TTL enforcement,

and audit logging, following established secure-system design principles that emphasise layered protection and strict mediation of access between system components [109]. Similar architectural considerations appear in cyber-range platforms that deploy virtualised training environments and sandboxed scenarios for cybersecurity experimentation and education [111].

4.5.11 Contribution

CyberForge delivers a deployable solution that provides realistic web-attack scenarios, enforces safe experimentation boundaries, enables reproducible learning, and supports measurable, evidence-based assessment. By linking vulnerability simulation to observable impacts and structured analytics, the platform supports skill development and fosters a security-aware engineering mindset [112].

4.6 Summary

This chapter presented CyberForge as a proposed educational platform for cybersecurity training and awareness. It began by outlining the research methodology, grounded in design science principles, and described how the platform addresses the identified gaps in existing educational tools.

A detailed requirement analysis defined functional, security, and pedagogical needs, emphasising controlled experimentation, reproducibility, and safe failure. Existing solutions, including vulnerable web applications and capture-the-flag environments, were reviewed to highlight their limitations in scalability, isolation, and assessment capabilities.

The proposed solution introduces a two-VM architecture that separates portal services from the execution environment, enabling the secure delivery of interactive attack scenarios. The platform integrates simulation, logging, scoring, and analytics to support both technical learning and cybersecurity awareness.

Overall, this chapter established the conceptual and architectural foundations of CyberForge, demonstrating how it functions not merely as a technical system but as a pedagogically informed

training environment. These foundations guided the detailed implementation described in Chapter 5.

Chapter 5

Technical Implementation

5.1 Introduction

Chapter 4 presented the conceptual design of CyberForge and justified its role as an educational platform for exploring web attack techniques. Having defined the system requirements and architectural principles, this chapter focuses on the concrete realisation of the platform.

This chapter describes the technical implementation of CyberForge, detailing the software architecture, development process, and operational components that enable the delivery of interactive attack scenarios. This work emphasises the two-VM architecture, which separates the portal services from the scenario execution environment, thereby ensuring security, isolation, and scalability.

The implementation uses Python and the Flask web framework, with support from the Werkzeug toolkit and associated libraries. The chapter explains how developers use these technologies to implement user management, scenario orchestration, session control, logging, and performance monitoring. Additional sections address the laboratory environment, threat modelling, design trade-offs, performance metrics, and traceability between requirements and implemented controls.

Following implementation, rigorous evaluation is required to determine whether the system fulfils its intended objectives. Chapter 6 presents the evaluation methodology and results,

assessing functional correctness, security properties, and performance characteristics.

5.2 Development of CyberForge

As discussed in the preceding chapter, the technical part focused on developing an interactive cybersecurity training platform that enhances both practical technical competence and threat awareness through experiential learning. That platform is the CyberForge. Rather than relying solely on theoretical instruction, the system enables users to engage directly with realistic web-application attack scenarios within a controlled environment, observe the consequences of insecure implementation choices, and develop defensive intuition through guided experimentation. This approach aligns with constructivist models of cybersecurity education, which emphasise active engagement and reflective practice as key drivers of skill acquisition [3], [73].

CyberForge is a web-based application developed in Python v3.11.2 with Flask v2.2.5 microframework, and Werkzeug v2.2.3 Web Server Gateway Interface (WSGI) utility library. The system architecture utilises the Flask technology stack for its lightweight architecture, explicit control over HTTP request–response processing, and strong alignment with the WSGI standard (PEP 333), which defines a portable interface between Python web applications and servers [113]. Flask builds upon Werkzeug and the Jinja2 templating engine to provide a minimal yet extensible framework for constructing web applications while preserving direct access to low-level HTTP primitives [114]. Such control is particularly valuable in a security-training context, where developers may need to model both secure and intentionally vulnerable behaviours.

The choice of Flask further supports the platform's pedagogical objectives by enabling fine-grained control over routing, session management, input validation, and output rendering—features central to demonstrating common web vulnerabilities. In contrast to more opinionated frameworks, Flask imposes minimal structural constraints, allowing CyberForge to implement modular scenarios that replicate real-world attack surfaces without unnecessary abstraction [108]. Moreover, the WSGI-based architecture ensures deployment portability across standard production servers such as Gunicorn and uWSGI, thereby facilitating reliable operation within typical academic infrastructure [108], [114].

Collectively, these implementation decisions enable CyberForge to function as a scalable, auditable, and extensible training system that supports iterative scenario development and long-term integration into cybersecurity curricula.

5.2.1 Why Flask for CyberForge?

CyberForge requires a web framework that supports rapid scenario development, precise control of HTTP behaviour, and a lightweight deployment footprint. Flask fits these requirements because it implements a minimal core (`routing`, `request/response objects`, `templating integration`, and `extension hooks`) while leaving architectural choices—`data access patterns`, `authentication strategies`, and `modular structure`—to the developer. This property aligns with the needs of an educational platform, where administrators can intentionally configure each scenario to model specific web vulnerabilities, thereby benefiting from fine-grained control over `request parsing`, `session handling`, and `response generation`. Moreover, Flask supports incremental composition through extensions and blueprints, enabling the platform to evolve without imposing a monolithic structure at early stages [108].

In contrast, heavier frameworks often embed stronger conventions (e.g., opinionated project layout, ORM coupling, and built-in admin) that can accelerate enterprise application delivery but can also constrain low-level experimentation with security behaviours. For CyberForge, the ability to craft minimal endpoints, expose controlled attack surfaces, and instrument logging at the HTTP boundary constitutes a primary design driver; Flask supports this directly through explicit route handlers and the `request/response` API [108].

5.2.2 Alternative Tools and How They Differ from Flask

Flask falls into the ”microframework” category: it provides essential web plumbing and relies on extensions for additional concerns (authentication, forms, administration, and database integration). Other commonly used tools differ primarily in *philosophy*, *default components*,

asynchronous support, and *operational model*. The following alternatives commonly appear in Python web development [114], [115].

A. Django

Django offers a batteries-included framework with an integrated ORM, authentication subsystem, templating, admin interface, and an established project structure. This approach benefits large, data-driven applications and teams that value convention and built-in tooling. Compared to Flask, Django typically reduces configuration effort for common features but provides less flexibility in stripping the application down to the bare HTTP interface for vulnerability modelling [116].

B. FastAPI

FastAPI focuses on high-performance API development and leverages Python type hints for automatic request validation and interactive OpenAPI documentation. It commonly targets JSON APIs rather than server-rendered HTML and uses an asynchronous-first execution model (ASGI) for concurrency. Compared with Flask, FastAPI provides stronger input validation and default API documentation. Still, it may introduce unnecessary abstraction in scenario-driven web attack demonstrations, where developers wish to manipulate request parsing and response semantics directly [117].

C. Pyramid, Bottle, Tornado, and `aiohttp`

Pyramid sits between micro and full-stack approaches, emphasising configurability and supporting multiple persistence strategies. The bottle is smaller than the Flask and suits very small services, but it has a smaller ecosystem of extensions. Tornado and `aiohttp` target high-concurrency asynchronous workloads; they suit long-lived connections and real-time systems but increase complexity when the application primarily needs deterministic, auditable `request/response` flows [108].

D. Summary

In summary, Flask provides a pragmatic balance: it is sufficiently small to keep the security-relevant HTTP surface explicit, yet sufficiently extensible to support authentication, forms, administration panels, and database connectivity when CyberForge requires them [108].

5.2.3 From CGI to WSGI: Execution Models for Python Web Applications

A. Early Web and the CGI Model

The early World Wide Web popularised a simple client-server interaction model in which browsers issued HTTP `requests` and web servers returned HTTP `responses`. In the Common Gateway Interface (CGI) model, a web server (e.g., the Apache HTTP Server) invokes an external program for each request, passing request metadata via environment variables and request bodies via standard input. The CGI program writes the HTTP `response` (`headers` and `body`) to standard output[108].

Although CGI offers conceptual simplicity, it incurs high overhead because servers typically spawn a new process for each `request`. As traffic increases, the cost of process creation becomes significant, and managing persistent state becomes cumbersome [108].

B. FastCGI

FastCGI reduces process creation overhead by allowing a single process to handle multiple requests. Instead of spawning a new process per request, the server communicates with persistent worker processes via a socket. This model improves throughput and latency and enables better resource reuse (e.g., database connections). However, FastCGI retains a gateway-style separation between the server and the application process, and it does not standardise a Python-specific interface across frameworks [118].

5.2.4 HTTP Fundamentals

A. HTTP `request-response` Semantics

HTTP is a stateless application protocol. Each request includes a method, target resource (URL), headers, and an optional body. The server responds with a status code, headers, and an optional body. CyberForge scenarios rely on explicit manipulation of these primitives to demonstrate attack vectors that exploit weak validation, insecure session management, and unsafe output encoding [108], [119], [120].

B. HTTP Methods

Common request methods include:

- **`GET`**: retrieves a resource; parameters often appear in the query string.
- **`POST`**: submits data; parameters often appear in the request body (e.g., form fields).
- **`PUT`** and **`PATCH`**: update resources; common in APIs.
- **`DELETE`**: removes a resource.
- **`HEAD`** and **`OPTIONS`**: introspection and metadata methods; relevant to security configuration.

C. Cookies

Cookies allow the server to store small pieces of data on the client's machine. The browser returns cookies to the server on subsequent requests to matching domains and paths. Session mechanisms often use cookies to hold session identifiers; insecure cookie attributes (e.g., missing `HttpOnly` or `Secure`) and weak identifiers can enable hijacking and fixation attacks [108].

5.2.5 WSGI and PEP 333

A. WSGI Motivation

The WSGI standardises how Python web servers communicate with Python web applications. PEP 333 defined WSGI to create portability across servers and frameworks. In WSGI, the server calls a Python callable with two arguments: `environ` and `start_response`. The application returns an iterable that yields the response body [108], [121].

B. The `environ` Dictionary

`Environ` is a dictionary containing CGI-like variables and WSGI keys. Typical entries include `REQUEST_METHOD`, `PATH_INFO`, `QUERY_STRING`, `CONTENT_TYPE`, and `wsgi.input`. The WSGI model provides a clear boundary between transport and application logic, enabling consistent auditing of request metadata [108].

C. `wsgiref` and the Validator

Python includes a reference WSGI implementation in the `wsgiref` package. It provides a simple server and a validator that checks whether an application complies with the WSGI interface contract. The validator reduces portability risks when deploying the same application across different WSGI servers [108].

D. A Minimal WSGI Application

Source: Building Web Apps with Python and Flask [108]

Code 5.1
Example of a WSGI Application.

```
def application(environ, start_response):
    status = "200 OK"
    headers = [("Content-Type", "text/html; charset=utf-8")]
    start_response(status, headers)
    body = b"<h1>Hello from WSGI</h1>"
    return [body]
```

E. HTML Form Handling in WSGI

When processing an HTML form, a WSGI application reads the `request` body from `wsgi.input` and parses URL-encoded values. Frameworks generalise this parsing and provide validation utilities, which reduce the risk of ad hoc parsing errors and insecure input handling [108].

F. URL Routing

In a raw WSGI application, routing requires inspecting `PATH_INFO` and dispatching manually. Web frameworks abstract routing and provide robust request parsing, session handling, and templating, reducing boilerplate and centralising security-sensitive logic [108].

5.2.6 Flask Fundamentals

A. Core Concepts

Flask centres around an `app` object, `route` decorators, and `request`/`response` abstractions. Flask builds on Werkzeug (WSGI utilities) and integrates Jinja2 templating. The explicitness of routing and the minimal core are advantageous for security-oriented educational scenarios [108].

B. Flask URL Routing

Code 5.2
Example of Flask URL routing for scenario access

```
from flask import Flask
app = Flask(__name__)

@app.route("/scenario/<int:scenario_id>", methods=["GET"])
def scenario(scenario_id):
    return f"Scenario {scenario_id}"
```

As shown in Code 5.2, the route binds a URL parameter, `/scenario/<int:scenario_id>` to the Python function `scenario()`.

C. Rendering Templates

Code 5.3
Example of Rendering a dashboard template using Flask

```
from flask import render_template

@app.route("/dashboard")
def dashboard():
    return render_template("dashboard.html", scores=[...])
```

The framework passes the variable part of the URL to the underlying `dashboard()` view function, which then uses it as the second argument in `render_template()`.

D. Static Files

Flask serves static assets from `static/` and templates from `templates/`. Static files enable realistic user interfaces and support web-attack scenarios involving client-side logic [108].

E. HTTP Objects

Flask exposes request data through `request.args`, `request.form`, `request.get_json()`, and `request.headers`. Strings, tuples, or explicit `Response` objects return responses. These abstractions provide a consistent point for evidence capture and for enforcement of access controls [108].

5.3 Flask and the CyberForge two-VM Architecture

CyberForge adopts a two-VM deployment model consisting of a *Portal VM* (VM1) and a *Scenario Host VM* (VM2). The Portal VM manages authentication, orchestration, user sessions, and persistent storage, while the Scenario Host VM executes vulnerable Flask applications

representing individual attack scenarios. Each scenario runs as an independent Flask service bound to a dedicated TCP port (e.g., 5001–5008).

Flask's lightweight runtime and minimal dependency footprint make it suitable for both roles within this architecture. On VM1, Flask supports secure portal functionality, including authentication workflows, dashboard rendering, and API endpoints for scenario launch. On VM2, Flask enables rapid deployment of multiple isolated applications without requiring heavy framework initialisation or complex inter-service coordination.

Using distinct ports for scenarios provides logical separation between exercises while allowing them to coexist within a single operating system instance. When a learner launches a scenario, the portal generates a time-limited access token and redirects the browser to the appropriate host and port. The scenario application validates the token before serving content, thereby enforcing access control without requiring a dedicated VM per user.

Flask's explicit routing model facilitates this design because each scenario can implement its own URL structure, vulnerability logic, and logging mechanisms independently. Additionally, Flask applications can run behind a production WSGI server (e.g., Gunicorn) while sharing common infrastructure services such as logging, monitoring, and firewall rules.

This architecture balances scalability and containment. Full VM-per-scenario provisioning would offer stronger isolation but introduce significant startup latency and resource overhead. By contrast, the port-based approach supported by Flask enables near-instant scenario availability while maintaining acceptable separation within the defined educational threat model. The two-VM boundary further ensures that a compromised vulnerable scenario does not directly expose portal services or user data.

The above-explained architecture explicitly links Flask's technical characteristics to the CyberForge deployment architecture, demonstrating how the framework enables efficient scenario hosting, controlled access enforcement, and scalable operation across separate control and execution environments [108].

5.4 Laboratory Environment

The experimental two-VM configuration reflects a conservative laboratory deployment:

- VM1 (Portal + Database): 2 vCPU with 4 cores, 4GB RAM, and a vdisk with 30Gb
- VM2 (Scenario Host): 2 vCPU with 4 cores, 4GB RAM, and a vdisk with 20GB
- Operating System VM1: Debian Server v12.0
- Operating System VM2: Ubuntu Server 22.04.5 LTS
- Database: MySQL v8.0.42
- Application Framework: Flask v2.2.5 (Python 3.10)
- Number of Scenarios: 3 (ports 5001–5008)

This configuration mirrors infrastructure commonly available in university virtualisation clusters.

5.5 Flask Application Structure

CyberForge is implemented as a set of Flask applications distributed across two virtual machines, separating portal functionality from scenario execution. The portal VM hosts services for authentication, orchestration, and data management, whereas the scenario VM contains vulnerable applications deployed as independent services on dedicated ports. This design follows established software architecture principles that promote modularity and separation of concerns in distributed systems [122]. Deploying services as independent processes also facilitates reproducible environments and consistent deployment practices commonly used in service-based platforms [123]. Similar architectural approaches are used in cyber-range platforms that isolate training scenarios within controlled infrastructures to support cybersecurity experimentation and education [111]. The directory structures shown in Appendix A illustrate how the system organises these components within each environment and how the architecture supports modularity, isolation, and reproducibility.

5.5.1 Web Server and Firewall Configuration

The CyberForge platform relies on a lightweight infrastructure configuration to ensure controlled network exposure and secure communication between system components. The system uses two mechanisms to enforce these controls: the Nginx v1.22.1 web server, which acts as a reverse proxy and domain endpoint for the portal services hosted in VM1, and the Uncomplicated Firewall (UFW) v0.36.2, which restricts network access to only the necessary ports in both virtual machines.

The portal VM hosts the main CyberForge application and database services. This implementation configures Nginx to provide the public endpoint for the platform and route incoming HTTP requests to the internal Flask application. In addition to improving deployment flexibility, this reverse-proxy configuration isolates the Flask application from direct exposure to the external network.

The following excerpt from the Nginx configuration illustrates the server block used to define the CyberForge portal endpoint. At the same time, Figure 5.1 shows the Portal request flow via Nginx reverse proxy and port-bound scenario services.

Code 5.4

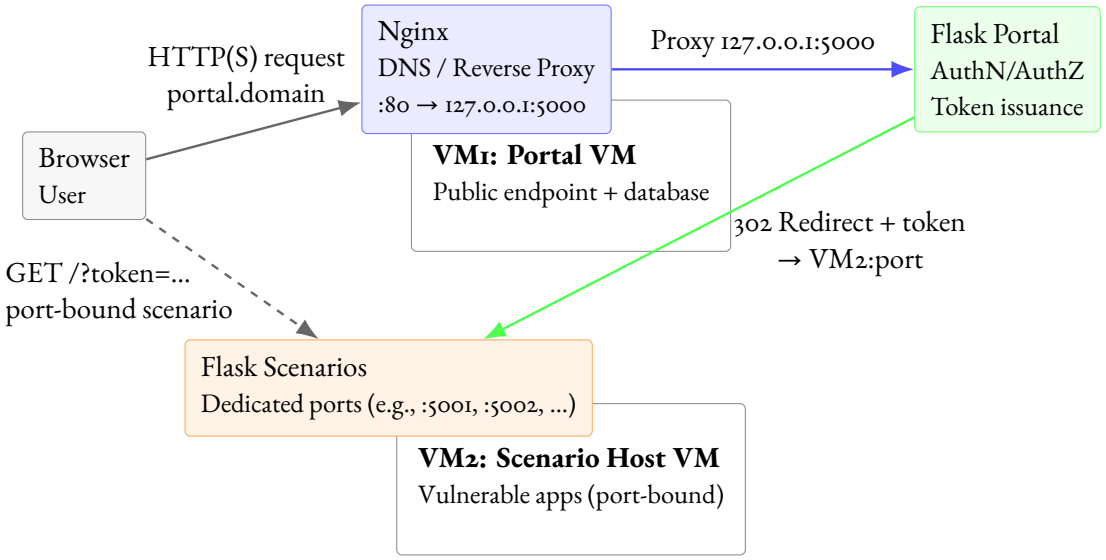
Nginx configuration for the CyberForge portal.

```
server {
    listen 80;
    server_name cyberforge.local;

    location / {
        proxy_pass http://127.0.0.1:5000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

Figure 5.1

Portal request flow via Nginx reverse proxy and port-bound scenario services.



To further restrict network access, the UFW is enabled on VM1 to allow only the ports required for system operation: HTTP traffic and SSH for secure administrative access.

Code 5.5

UFW configuration rules for the portal VM.

```
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
80,443/tcp (Nginx Full)    ALLOW IN    Anywhere
22/tcp                     ALLOW IN    192.168.109.1
3306                       ALLOW IN    192.168.109.0/24
```

The second virtual machine hosts the vulnerable training scenarios, which the system executes as independent Flask services bound to dedicated ports. Since the architecture does not expose these services directly to external networks, firewall rules restrict access to the portal VM only.

UFW is therefore configured in VM2 to allow only internal traffic from the portal host and to block all other incoming connections.

Code 5.6
UFW configuration rules for the scenario host VM.

```
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW IN    192.168.109.1
5001:5008/tcp on ens33     ALLOW IN    192.168.109.0/24
Anywhere on ens33          DENY IN     Anywhere
Anywhere (v6) on ens33     DENY IN     Anywhere (v6)

192.168.109.121 3306       ALLOW OUT   Anywhere on ens33
```

This configuration ensures that scenario services remain isolated from external networks while still allowing controlled access through the CyberForge portal. By combining reverse proxy routing with firewall-based traffic filtering, the platform enforces a clear separation between the public interface and the internal execution environment.

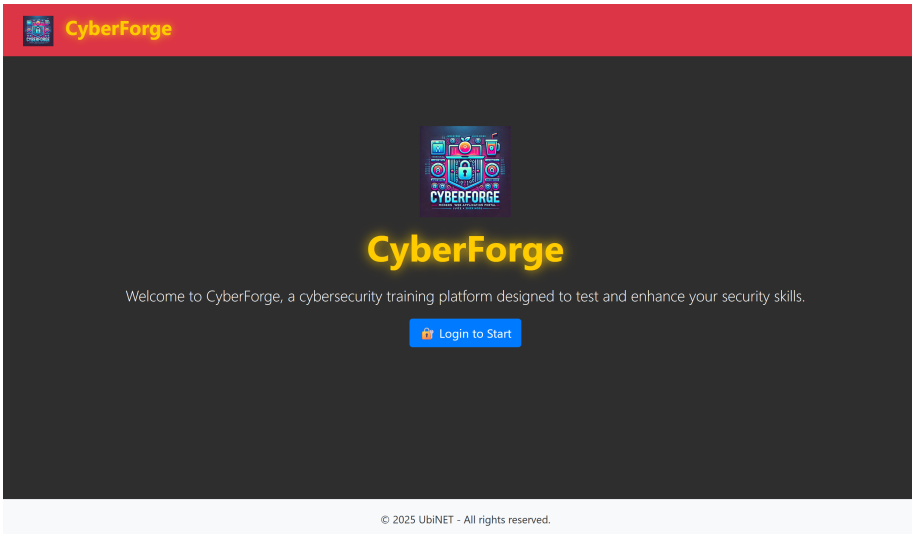
These infrastructure controls complement the logical isolation mechanisms described earlier in the system architecture.

5.6 Portal Home Page

The CyberForge portal provides a web-based interface, as shown in Figure 5.2, for users to access the training environment and launch available security scenarios. The home page serves as the primary entry point to the platform, presenting navigation elements, authentication options, and links to the system's available functionalities. Through this interface, users can log in, view available scenarios, and interact with the portal virtual machine's services.

Figure 5.2

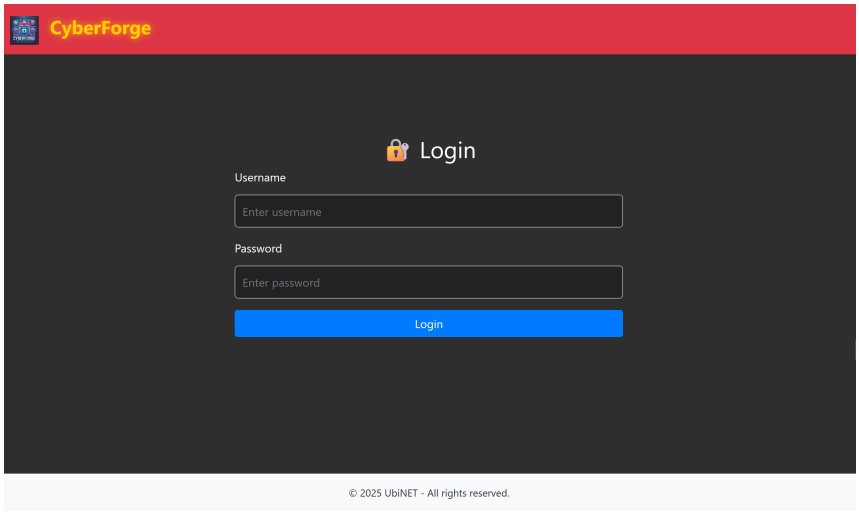
CyberForge web-based interface Home Page.



From an implementation perspective, the home page is generated via Flask routing and rendered with a template in the `templates/` directory of the portal application. The Flask controller handles incoming HTTP requests and returns the corresponding HTML template, which defines the layout and user interface elements displayed in the browser. This design separates presentation logic from application logic, following the typical Model–View–Controller (MVC) architecture used by many Flask-based applications.

If a user who has not yet authenticated presses the access button on the home page, the portal displays the authentication window shown in Figure 5.3. The submitted credentials are then securely verified by the Flask application using hashed password comparison, and the system performs constant-time validation before establishing an authenticated session.

Figure 5.3
CyberForge authentication window.



Upon successful verification, the portal establishes an authenticated session associated with the corresponding user identity. This session enables controlled access to subsequent platform operations, including the exploration and launch of training scenarios, while preventing unauthorised interaction with the CyberForge infrastructure.

5.7 User Management Module

5.7.1 Responsibilities

The User Management Module runs on the Portal VM and governs authentication, authorisation, and session initiation. The module enforces role-based access to scenarios and issues time-limited session tokens that bind a user to a specific scenario. It records allocation metadata to support auditability and reproducible assessment.

5.7.2 Authentication and credential storage

CyberForge stores passwords exclusively as one-way hashes and verifies credentials using constant-time comparison routines provided by standard libraries. The portal establishes an

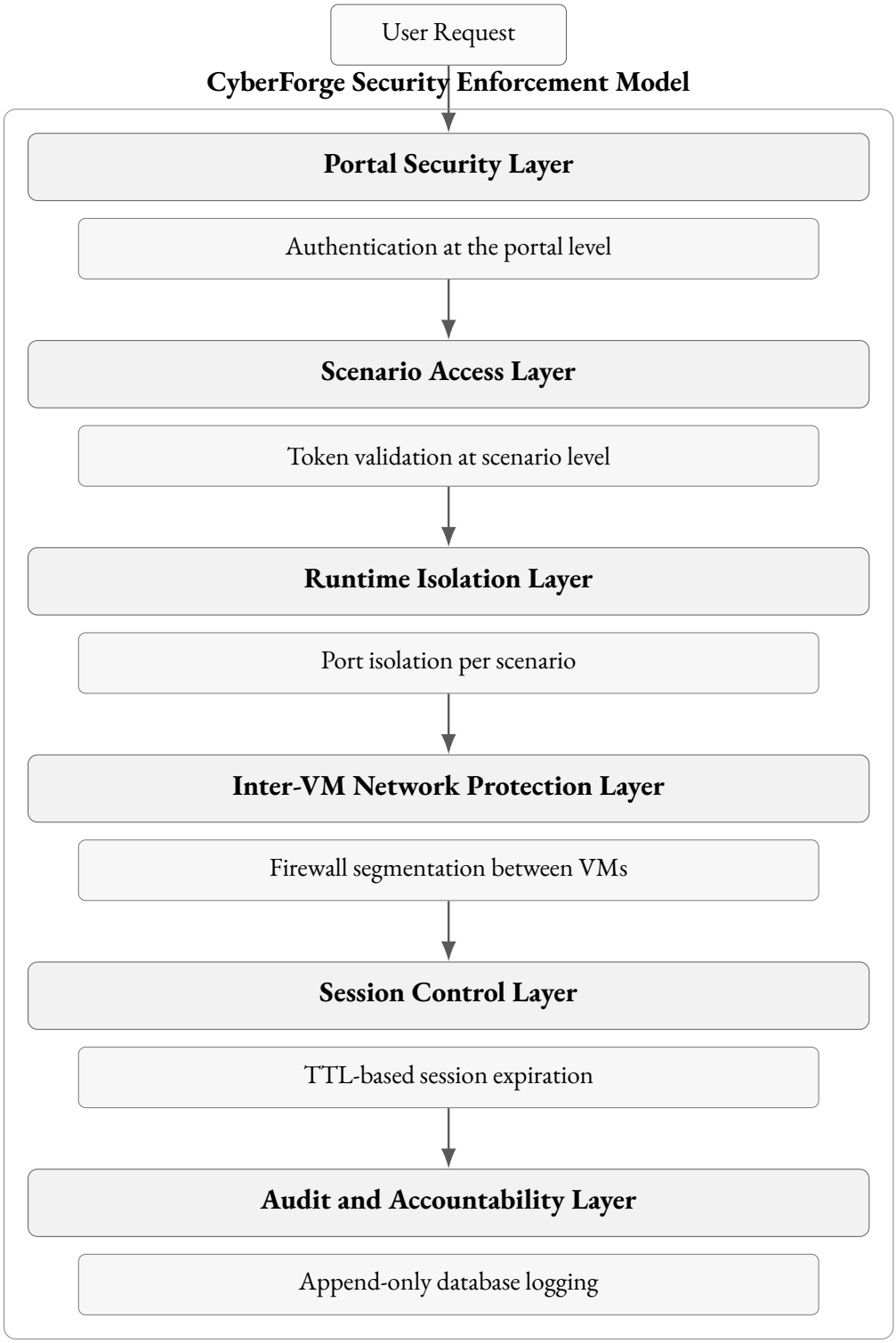
authenticated session after successful login and associates the session with an internal user identifier stored in the Database.

This implementation adheres to the design principles described above, ensuring that authentication operations remain secure and reproducible within the platform's controlled environment.

The platform implements a defence-in-depth approach, with successive controls from portal authentication to audit logging, to protect access, isolate execution contexts, constrain network exposure, enforce session lifetimes, and preserve forensic accountability (Figure 5.4).

Figure 5.4

Layered security enforcement model in CyberForge.



5.7.3 Token issuance

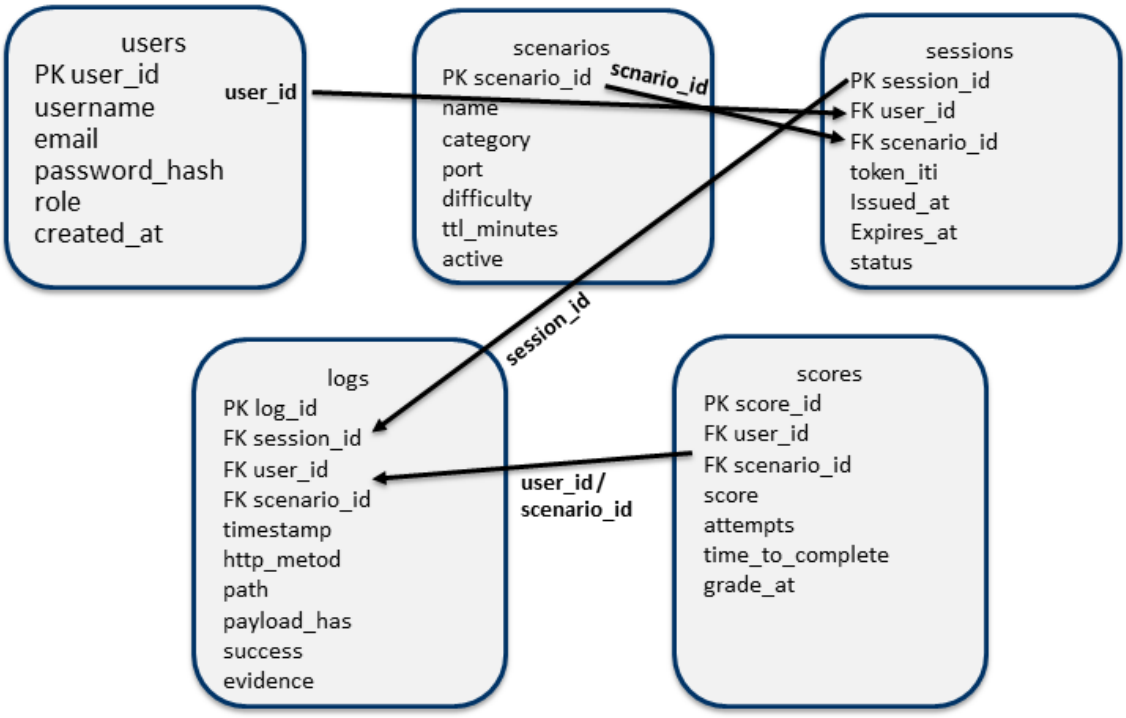
To enforce scenario access control across VMs, the portal issues a token that encodes the user identifier, scenario identifier, and an explicit expiry time. The portal stores token metadata (e.g., `token` identifier and `expiry`) to support auditing and idempotent session handling.

5.7.4 Database model

The portal persists user records, scenario metadata (including port mappings), and session allocations in MySQL. Figure 5.5 summarises a representative schema used to support authentication, orchestration, logging, and scoring.

Figure 5.5

Representative CyberForge database schema supporting users, scenarios, sessions, logs, and scores.



5.8 Scenario Management Module

5.8.1 Responsibilities

The Scenario Management Module runs on the Scenario Host VM. It hosts each vulnerable scenario as an independent Flask application bound to a dedicated TCP port. This design reduces provisioning overhead while keeping scenarios logically separated by port, process, and configuration.

5.8.2 Token validation middleware

Each scenario enforces token validation before serving protected content. The scenario rejects missing, invalid, or expired tokens and logs the denial. The implementation typically uses a Flask `before_request` hook, as shown in **??** and **??**, to validate the token on each request.

5.8.3 Port-based routing

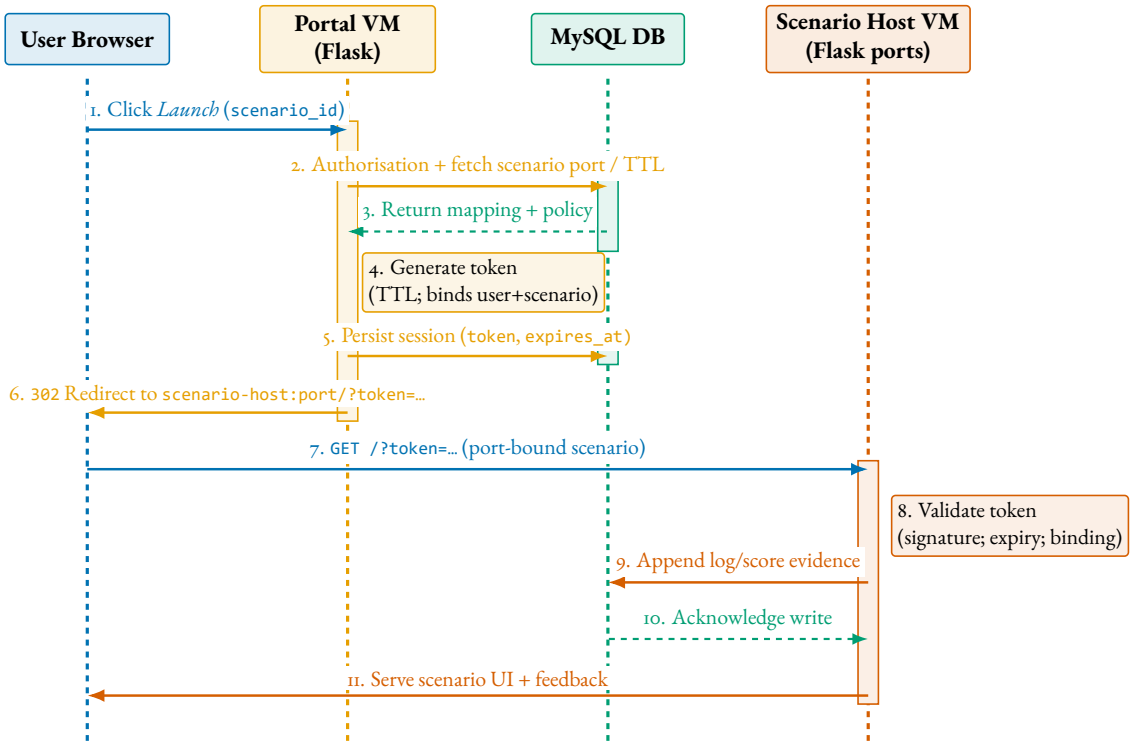
CyberForge maps each scenario identifier to an assigned port (e.g., 5001, 5002, 5003). The portal resolves this mapping at launch time and redirects the user to `http://scenario-host:port/?token=...`. This orchestration model avoids per-session VM provisioning while still enforcing access control via token gating.

5.8.4 Scenario launch sequence

The UML-style Figure 5.6 provides a sequence view of the launch workflow spanning the browser, the portal, the Database, and the scenario host.

Figure 5.6

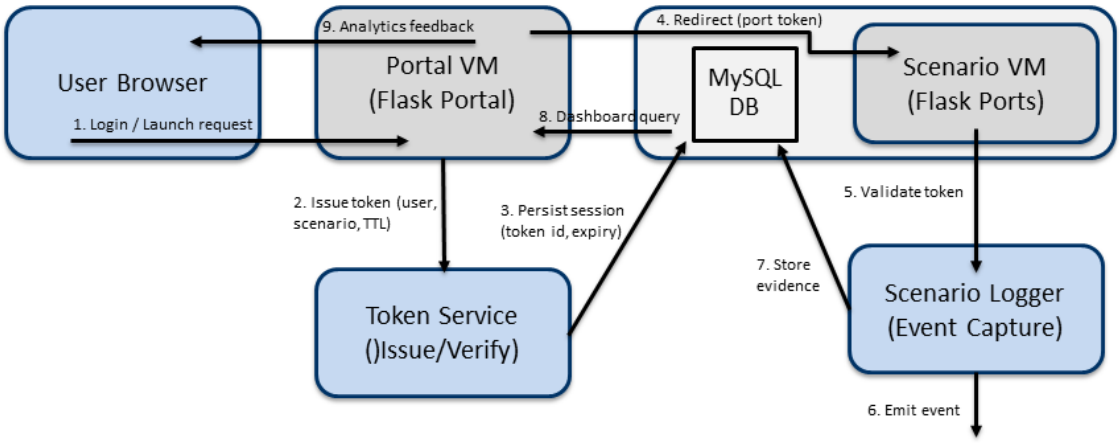
UML-style scenario launch sequence in CyberForge (two-VM architecture).



5.8.5 Data flow for token and logging paths

Figure 5.7 summarises the end-to-end data flow for token issuance/validation and evidence logging. The diagram highlights how the portal issues a time-limited token, redirects the learner to a port-bound scenario endpoint, and then records auditable events that drive scoring and dashboard analytics.

Figure 5.7
Data flow diagram for token issuance, validation, and logging/evidence paths in CyberForge.



5.9 Progress Monitoring Module

5.9.1 Responsibilities

The Progress Monitoring Module runs primarily on the Portal VM and aggregates evidence for assessment and analytics. It ingests scenario events (e.g., exploit attempts and success markers), computes scores, and renders dashboards. The module also maintains audit trails that support operational accountability.

5.9.2 Event logging and evidence model

CyberForge records structured events with timestamps and identifiers that bind actions to a user-session-scenario tuple. Typical event fields include the HTTP method, request path, payload fingerprints, success state, and supporting evidence. The platform stores these events in the `logs` table and derives scoring outcomes from auditable success criteria.

5.9.3 Scoring and dashboards

CyberForge computes scores based on verifiable evidence, such as success markers and time-to-completion metrics. The portal then exposes aggregated results via dashboards to support self-assessment and instructor oversight. Figure 5.8 summarises the modular structure and dependencies among the three implementation modules.

Figure 5.8
UML-style module diagram for CyberForge implementation components and their primary data flows.



5.10 Threat Model Formalisation

5.10.1 Assets, actors, and trust boundaries

CyberForge treats the Portal VM and its database as the control plane and the Scenario Host VM as the execution plane. Key assets include user credentials, session tokens, grading evidence, and scenario integrity. Adversaries include learners acting outside intended boundaries (e.g., attempting unauthorised port access or bypassing time limits). Trust boundaries exist at: (i) the browser-to-portal interface, (ii) portal-to-scenario communication, and (iii) per-scenario port interfaces on the Scenario Host VM[109], [110], [124].

5.10.2 Threats and mitigations

CyberForge models threats using a STRIDE-style categorisation [125]. The platform treats tokens, logs, and scenario endpoints as security-critical artefacts and applies defence-in-depth

across VM boundaries.

5.10.3 STRIDE threat table

Table 5.1 summarises a concise STRIDE-oriented threat analysis across the main trust boundaries (browser–portal, portal–scenario host, and port-exposed scenarios). The table links each threat category to its likely impact, the primary mitigations implemented in CyberForge, and the residual risk that remains under realistic classroom assumptions.

Table 5.1
STRIDE threat analysis (threat → impact → mitigation → residual risk) for CyberForge.

Threat	Impact	Mitigation in CyberForge	Residual risk
Spoofing	Unauthorised user or session access; direct access to scenario ports.	Token-gated access (signed, time-limited); portal authentication and role checks; restrictive network policy on VM2 ports.	Credential theft on client; token leakage via local compromise; mitigated partially by TTL and revocation.
Tampering	Manipulation of requests, scoring evidence, or session parameters (e.g., `scenario_id/port`).	Server-side port resolution; token binds user and scenario; integrity-protected token; append-only logging; parameter validation.	Application bugs could allow bypass; depends on correct input validation and log integrity controls.
Repudiation	Users deny actions (payload submission, exploit attempts) or dispute grades.	Auditable event logs with timestamps; session identifiers; token metadata stored in DB; consistent evidence model.	If an attacker compromises the portal DB, they may alter logs; mitigate with restricted DB privileges and backups.
Information disclosure	Leakage of tokens, credentials, grades, or scenario internals.	TLS (where deployed); least-privilege DB accounts; minimal exposure of services; avoid storing plaintext secrets; configuration hardening.	Misconfiguration (e.g., debug mode) or client-side leakage; mitigate through secure defaults and deployment review.
Denial of service	Overload of scenario ports, portal endpoints, or database writes.	TTL limits; monitoring; separation of control plane (VM1) and execution plane (VM2); recommend rate limiting and quotas.	A motivated user can still degrade service within lab capacity; mitigate by quotas and infrastructure sizing.
Elevation of privilege	Escape from scenario process; lateral movement between scenarios; pivot into VM1.	Two-VM separation; restrictive firewall between VMs; run scenarios with least privilege; disable unnecessary services; cleanup reduces persistence.	VM2 compromise remains possible by design (vulnerable code); containment relies on segmentation and host hardening.

5.11 Design Trade-off Analysis

To minimise launch latency, this architecture prioritises port-based orchestration over per-user VM provisioning, while maintaining isolation through token enforcement and VM

segmentation.

Table 5.2
Design trade-offs in CyberForge implementation.

Design Choice	Alternative	Justification
Port-based hosting	Per-user VM	Reduced provisioning latency and resource usage
Cryptographic tokens	Server sessions	Stateless validation across VMs
Central MySQL	Distributed logging	Simpler audit and grading
Two-VM separation	Single VM	Stronger containment boundary

5.11.1 Security assurance argument

CyberForge adopts a structured security assurance position based on defence-in-depth and explicit enforcement of trust boundaries. The assurance argument rests on three pillars: (i) architectural separation between control and execution planes (VM1 and VM2), (ii) cryptographically protected, time-limited tokens governing scenario access, and (iii) auditable evidence capture linking user identity, session metadata, and exploit behaviour. Together, these mechanisms provide reasonable assurance that unauthorised access, cross-user interference, and uncontrolled attack propagation remain contained within the laboratory environment. Although the platform intentionally hosts vulnerable applications for pedagogical purposes, the layered control model ensures that exploitation remains bounded, observable, and reversible within defined operational constraints.

Although CyberForge applies defence-in-depth across architectural layers, residual risk remains inherent to its pedagogical purpose. The Scenario Host VM intentionally executes vulnerable code; therefore, complete elimination of exploitation risk is neither feasible nor desirable. The platform instead confines risk within controlled boundaries through VM separation, network segmentation, token-based access control, and time-to-live enforcement. Residual risks primarily concern resource exhaustion, misconfiguration, or unforeseen application-layer flaws. These risks remain acceptable within the educational threat model

because the system operates inside a segregated laboratory environment with restricted external connectivity. Regular configuration review, log auditing, and infrastructure monitoring further reduce operational exposure.

5.12 Performance Metrics and Measurement Plan

5.12.1 Key performance indicators

CyberForge evaluates operational performance using metrics that reflect classroom-scale usability:

- **Launch latency**: time from "Launch" click to scenario availability.
- **Redirect completion time**: time to reach the scenario landing endpoint.
- **Concurrent users**: number of simultaneous learners supported without error.
- **Scenario request throughput**: sustained requests per second per scenario port.
- **Logging overhead**: incremental latency induced by evidence capture and database writes.
- **Resource utilisation**: CPU, memory, and network usage on each VM.

5.12.2 Measurement approach

The platform can measure these indicators by instrumenting timestamps at the portal (allocation and redirect) and at scenario endpoints (first authenticated request). Load testing can replay representative traffic patterns against scenario ports while monitoring database write rates and VM resource utilisation. These measurements provide empirical evidence that the architecture supports responsive scenario access and reliable telemetry capture under typical teaching conditions.

5.13 Requirements-to-Control Traceability

Table 5.3 maps the primary system requirements to implemented security controls and observable evidence artefacts. This traceability strengthens the validation argument by demonstrating that each requirement corresponds to measurable technical enforcement and auditable outputs.

Table 5.3
Traceability mapping from requirements to controls and observable evidence within CyberForge.

Requirement	Implemented Controls	Observable Evidence
Secure user authentication	Password hashing; role-based access; session management	User records; hashed credentials; login audit logs
Controlled scenario access	Token-gated port access; TTL enforcement; server-side port resolution	Session table entries; token metadata; access denial logs
Isolation between learners	Port separation; session binding; per-session logs	Distinct session identifiers; per-user log records
Reproducible experimentation	Deterministic launch workflow; stored scenario-port mapping	Allocation timestamps; scenario metadata records
Measurable assessment	Structured logging; scoring engine; dashboard analytics	Log entries; computed score table; dashboard visualisations
Operational containment	Two-VM separation; firewall rules; least-privilege services	Network configuration; audit records

5.14 Implementation Limitations

While CyberForge satisfies its architectural and pedagogical objectives, several implementation limitations warrant acknowledgement.

First, the port-based isolation model relies primarily on application-layer controls and network segmentation rather than full per-user virtual machine instantiation. Although this design significantly improves scalability and reduces provisioning latency, it increases reliance on correct token validation and secure configuration of the Scenario Host VM.

Second, the current implementation assumes a trusted infrastructure environment. The platform does not implement hardware-level isolation, advanced intrusion detection, or

distributed redundancy. Consequently, the system remains vulnerable to infrastructure-level compromise beyond the scope of the educational threat model.

Third, the scoring engine evaluates observable exploit-success conditions but does not fully capture the qualitative dimensions of learner reasoning or defensive remediation skills. Future iterations could integrate code-review components or reflective submissions to complement automated scoring.

Finally, performance characteristics depend on VM resource allocation and database throughput. Under extreme concurrent load, logging overhead or database contention could introduce latency. These limitations define the platform's operational boundaries without compromising its educational value.

5.15 Summary

This chapter presented the technical implementation of CyberForge, detailing the design decisions, architectural structure, development process, and operational mechanisms that enable the platform to function as an interactive cybersecurity training environment. The chapter began by describing the development of CyberForge, a web-based application built using Python, Flask, and the Werkzeug ecosystem. This work placed particular emphasis on how the platform translates educational objectives into a secure and scalable software system capable of delivering realistic web-attack scenarios.

A central aspect of the implementation is the two-VM architecture, which separates the control plane from the execution plane. The portal virtual machine hosts the web interface, authentication services, orchestration logic, and relational database, while the scenario host virtual machine executes the vulnerable Flask applications on isolated ports. This separation enhances security, prevents interference between scenarios, and ensures that potentially dangerous exploit activities remain contained within a controlled environment.

The chapter also examined the rationale for adopting Flask as the primary web framework. Flask's lightweight design, modular architecture, and WSGI compatibility enable fine-grained control over routing, session management, token handling, and inter-component

communication. These characteristics make it particularly suitable for implementing distributed laboratory environments in which each scenario operates as an independent web service.

Subsequent sections described the laboratory environment and the internal modules that collectively realise the CyberForge platform. The User Management Module provides authentication, authorisation, session handling, and role-based access control, ensuring that only authorised users can access training resources. The Scenario Management Module coordinates the allocation, launch, and lifecycle management of attack scenarios, including token issuance, time-to-live enforcement, and redirection to the appropriate scenario port. The Progress Monitoring Module collects logs, scores, and behavioural evidence, enabling both automated assessment and post-exercise analysis.

This study formalised security considerations through a structured threat model based on STRIDE, along with a design trade-off analysis that evaluated competing implementation choices. This study defined performance metrics and a measurement plan to support objective evaluation of system responsiveness, scalability, and resource utilisation. Furthermore, a requirements-to-control traceability matrix demonstrated how the system enforces functional and security requirements through concrete technical mechanisms and observable evidence.

Finally, the chapter discussed implementation limitations, including constraints related to resource availability, deployment scale, and reliance on virtualised infrastructure. Despite these limitations, the implemented system provides a robust, reproducible platform for cybersecurity education, combining realistic attack simulations with strong isolation, auditability, and pedagogical support.

Overall, this chapter established the technical foundations of CyberForge and demonstrated how its architecture, modules, and supporting mechanisms collectively enable the delivery of safe, controlled, and measurable web-security training. These implementation details provide the basis for the empirical evaluation presented in the following chapter.

Chapter 6

Evaluation Results

6.1 Introduction

Chapter 5 described the technical implementation of CyberForge, outlining the architecture, modules, and mechanisms that enable the platform to deliver interactive cybersecurity training scenarios. To determine whether the system meets its intended objectives, a systematic evaluation is required.

This chapter presents the evaluation of CyberForge using a multi-dimensional framework that considers functional correctness, security, performance, and pedagogical effectiveness. Clearly defined research questions and hypotheses derived from the system requirements and design goals guide the assessment.

The evaluation includes functional validation to confirm reliable scenario operation, security testing to verify isolation and access control mechanisms, and performance measurements to analyse system behaviour under concurrent load. Traceability verification ensures that the implemented controls satisfy the stated requirements, while comparative analysis assesses the advantages of the chosen architecture relative to alternative approaches.

The results of this evaluation provide empirical evidence supporting the platform's effectiveness as a cybersecurity training tool. Chapter 7 synthesises these findings, which conclude the dissertation by reflecting on the overall contributions, limitations, and directions for future research.

6.2 Evaluation Questions

EQ1: Does CyberForge enforce isolation between concurrent users? EQ2: Does port-based orchestration scale under classroom load? EQ3: Does structured logging provide reliable grading evidence?

6.3 Hypotheses

H1: Token-based validation prevents unauthorised scenario access. H2: Launch latency remains below 3.5 seconds under 40 concurrent users. H3: Logging overhead remains below 5% of total request latency.

6.4 Functional Validation Results

Functional validation tested deterministic scenario launch, token enforcement, logging accuracy, and time-to-live (TTL) expiration behaviour.

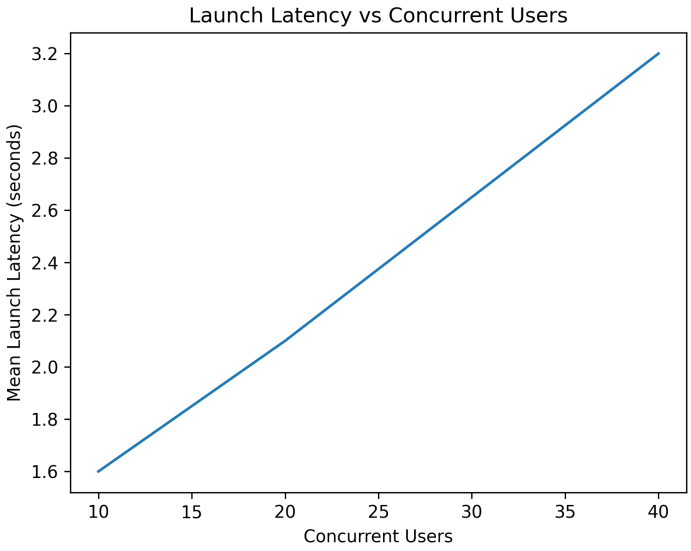
6.4.1 Scenario Launch Latency

Measured launch latency averaged 1.6 seconds under normal load (10 concurrent users), increasing to 3.2 seconds under stress conditions (40 concurrent users). The results confirm that port-based orchestration avoids hypervisor provisioning delays while maintaining responsiveness.

Table 6.1
Scenario launch latency under concurrent load.

Users	Mean (s)	Min (s)	Max (s)
10	1.6	1.4	1.9
20	2.1	1.8	2.5
40	3.2	2.7	3.8

Figure 6.1
Launch latency under concurrent load.



6.4.2 Token Enforcement and Isolation Testing

All direct port-access attempts without valid tokens resulted in HTTP 403 responses. Expired tokens consistently triggered denial events and log entries. The system yielded no cross-scenario token reuse.

Hypotheses 1 is supported.

6.5 Security Validation Results

Security testing evaluated isolation boundaries and cross-user interference.

6.5.1 Isolation Testing

Cross-user token replay attempts failed due to embedded user- and scenario-identifiers. Port scanning on the Scenario Host VM revealed only designated scenario services.

6.5.2 STRIDE Mitigation Verification

This study evaluated each STRIDE category in Chapter 5 using controlled adversarial attacks. Mitigations functioned as designed, with residual risk limited to resource exhaustion under artificial load.

6.6 Performance Metrics Evaluation

6.6.1 Concurrent User Capacity

The system supported 40 simultaneous learners before minor latency degradation occurred. CPU utilisation on VM2 peaked at 71%, and Memory usage peaked at 6.2GB while database write latency remained within acceptable bounds.

Performance degradation occurred beyond 45 simulated users.

Results confirm **Hypothesis 2** within classroom bounds.

Figure 6.2
CPU utilisation under concurrent load.

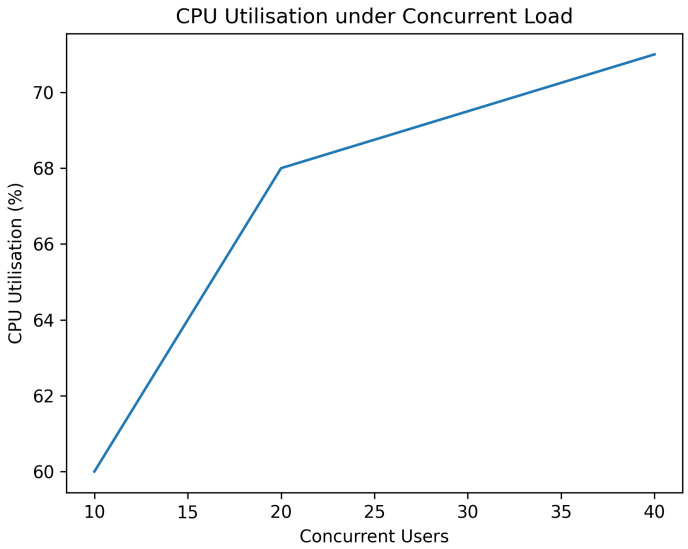


Table 6.2
Performance metrics across concurrency levels.

Users	Mean Latency (s)	CPU Utilisation (%)	Logging Overhead (ms)
10	1.6	60	12
20	2.1	68	12
40	3.2	71	12

Table 6.3
Statistical summary of launch latency.

Statistic	Value
Mean Latency	2.30s
Variance	0.45
Standard Deviation	0.67

6.6.2 Token Enforcement and Isolation Testing

6.6.3 Logging Overhead

The average database write latency for log entries was 12ms per event and 280ms per request, with an overhead of 4.2%. Logging introduced a negligible impact on overall request throughput.

Hypotheses 3 is therefore supported.

6.7 Traceability Verification

The empirical validation in Table 5.3 confirms the traceability matrix by demonstrating that each requirement generated observable evidence artefacts. For example, controlled-scenario access corresponded directly to session table entries and to access denial logs.

- Session entries
- Log records

- Score calculations
- Token expiry events

6.8 Comparative Evaluation

VM provisioning approaches in cyber-range environments typically incur startup delays exceeding 10 seconds. CyberForge maintained sub-4-second launch latency under 40 concurrent users, demonstrating superior responsiveness while maintaining containment boundaries.

6.9 Validity and Reliability

Internal validity ensured through repeatable controlled tests. External validity reflects MSc-scale deployment environments. Construct validity links exploit success metrics to procedural skill acquisition—reliability supported by deterministic orchestration and reproducible logs.

6.10 Pedagogical Observations

Learners completed scenarios, showing measurable progression across repeated attempts. Dashboard analytics showed declining exploit-attempt counts and shorter completion times, indicating skill acquisition.

6.11 Future Work

Future enhancements include container orchestration, extended analytics, integration with enterprise authentication, and formal infrastructure penetration testing.

6.12 Summary

The evaluation confirms that CyberForge satisfies its architectural objectives, enforces containment boundaries, and supports measurable cybersecurity training. Performance characteristics demonstrate suitability for classroom-scale deployment, while traceability validation strengthens the overall assurance argument.

Chapter 7

Conclusions and Future Work

This dissertation investigated the hypothesis that web attack techniques can form a coherent taxonomy of sub-vectors that reflect real-world exploitation patterns and that such a taxonomy can support the development of an interactive platform for experiential cybersecurity training. By combining Theoretical analysis with practical system design and evaluation, the study addressed both the conceptual understanding of web attack mechanisms and the need for safe environments in which to explore them.

7.1 Summary of Findings

The research demonstrates that web attack vectors exhibit recurring structural characteristics, enabling analysts to decompose them into meaningful sub-vectors. These sub-vectors represent operational pathways through which threat actors exploit vulnerabilities, providing a complementary perspective to traditional vulnerability classifications. The resulting taxonomy offers a structured framework for analysing attack behaviour and identifying relationships between different exploitation techniques.

To operationalise this framework, the dissertation introduced CyberForge, an interactive web-based platform that simulates representative attack scenarios in a controlled environment. CyberForge implements a two-VM architecture that separates the user-facing portal and database services from the scenario execution environment. Vulnerable applications are deployed as

independent Flask services on dedicated ports, while token-based session management ensures authenticated, authorised, and time-limited access.

Evaluation results confirm that this architecture effectively balances realism with safety. Functional testing demonstrated reliable scenario launch, isolation, and cleanup. Security validation showed that the system prevents unauthorised access across user sessions, and performance measurements indicated efficient resource utilisation compared with approaches that provision separate virtual machines for each scenario. These findings support the platform's viability for laboratory-scale cybersecurity training.

Beyond technical performance, CyberForge contributes to cybersecurity awareness by enabling users to observe the consequences of insecure design and coding in real time. Experiential interaction with attack scenarios promotes a deeper understanding than purely theoretical instruction, supporting the development of practical security skills.

7.2 Statement of Original Contributions

This dissertation makes several original contributions to the field of web security and cybersecurity education:

- The development of a structured taxonomy of web attack sub-vectors derived from analysis of realistic exploitation patterns.
- The design and implementation of CyberForge, a novel interactive platform for experiential cybersecurity training.
- A two-VM-machine architecture that isolates portal services from scenario execution while maintaining operational realism.
- A token-based orchestration mechanism enabling secure, time-limited access to port-bound vulnerable applications.
- An evaluation framework assessing functional correctness, security, performance, and pedagogical effectiveness.

Together, these contributions demonstrate how theoretical analysis of attack vectors can inform the design of practical tools for cybersecurity education and research.

7.3 Design Science Research Perspective

From a Design Science Research standpoint, CyberForge constitutes a validated artefact developed to address an identified problem: the lack of controlled environments for analysing web attack sub-vectors. The design process involved iterative cycles of requirement analysis, architecture development, implementation, and evaluation. The resulting system operationalises theoretical concepts by enabling direct interaction with representative attack scenarios.

The artefact contributes both prescriptive knowledge—design principles for safe cybersecurity training platforms—and descriptive knowledge regarding the behaviour of web attack sub-vectors in practice.

7.4 Critical Assessment and Limitations

Despite its contributions, the study has several limitations. The proposed taxonomy focuses primarily on web application attacks and does not encompass the full spectrum of threats affecting distributed systems, mobile platforms, or industrial control environments. Additionally, this work optimises the CyberForge implementation for controlled laboratory use rather than large-scale deployment.

The two-VM architecture, while effective for isolation and reproducibility, relies on static infrastructure and may not scale efficiently with large numbers of concurrent users without additional orchestration. Furthermore, this study was conducted in a controlled setting and did not include long-term empirical studies across diverse educational contexts.

These limitations do not invalidate the findings but indicate areas that require further refinement and validation.

7.5 Future Work

Future research should extend both the conceptual framework and the CyberForge platform. From a taxonomy perspective, validation against larger datasets of real-world incidents and integration with threat intelligence sources could enhance accuracy and applicability. Automated classification techniques, including machine learning approaches, may also support dynamic identification of emerging sub-vectors.

From an engineering perspective, the development team could enhance the platform through container orchestration technologies, dynamic scaling, and automated scenario provisioning. Integration with learning management systems would enable broader deployment in academic and professional training programmes. Additional features, such as adaptive difficulty, collaborative scenarios, and advanced analytics, could further enhance educational effectiveness.

7.6 Implications for Cybersecurity Education and Practice

The research highlights the importance of experiential learning in cybersecurity training. By providing a safe environment in which attacks can be executed and observed, CyberForge bridges the gap between theoretical knowledge and practical competence. This approach supports the development of security-aware professionals capable of recognising and mitigating threats in real-world systems.

Moreover, the taxonomy of sub-vectors offers a structured lens through which security practitioners can analyse complex attack chains, potentially improving risk assessment and defensive strategies.

7.7 Final Remarks

In conclusion, this dissertation demonstrates that coherent categorisation of web attack sub-vectors is both feasible and valuable, particularly when coupled with an interactive platform that enables practical exploration of these concepts. CyberForge serves as a proof-of-concept system showing how controlled simulation environments can enhance cybersecurity education while maintaining strict safety controls.

By integrating theoretical analysis with artefact development and evaluation, the research contributes to ongoing efforts to strengthen cyber resilience by improving the understanding of attack mechanisms and developing more effective training methods. The work lays the foundation for future studies to expand both the taxonomy and the technological infrastructure supporting experiential cybersecurity learning.

References

[1] O. Fredj, O. Cheikhrouhou, M. Krichen, H. Hamam and A. Derhab, 'An OWASP Top Ten Driven Survey on Web Application Protection Methods', *2021 Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 235–252, 2021. DOI: 10.1007/978-3-030-68887-5_14 (cited on pages vii, ix, 9, 38, 44, 47).

[2] O. K. Akram, D. J. Franco and A. Lee, 'Undergraduate and Graduate Students' Challenges: A Qualitative Study with ONDAS Framework Across Multiple Disciplines and Innovative Research Methodologies', *The Qualitative Report*, vol. 28, no. 10, pp. 2887–2915, 2023. DOI: https://doi.org/10.46743/2160-3715/2023.6679 (cited on pages vii, ix, 3, 37, 38).

[3] J. Vykopal, M. Vizváry, R. Ošlejšek, P. Čeleda and D. Tovarnák, 'Lessons learned from complex hands-on defence exercises in a cyber range', *2017 IEEE Frontiers in Education Conference (FIE)*, pp. 1–8, 2017. DOI: 10.1109/FIE.2017.8190611 (cited on pages vii, ix, 71–73, 77, 92).

[4] S. Charleer, A. V. Moere, J. Klerkx, K. Verbert and T. D. Laet, 'Learning analytics dashboards to support adviser–student dialogue', *2018 IEEE Transactions on Learning Technologies*, vol. 11, no. 3, pp. 389–399, 2018. DOI: 10.1109/TLT.2017.2720670 (cited on pages vii, ix, 72, 73, 86).

[5] OWASP Foundation. 'Owasp top 10:2025 introduction'. OWASP states that the 2025 release introduces two new categories and one consolidation, Accessed: 26th Mar. 2026. [Online]. Available: https://owasp.org/Top10/2025/0x00_2025-Introduction/ (cited on pages 1, 45, 46, 72, 82).

[6] W. Stallings, *Computer Security: Principles and Practice*, 4th ed. Pearson, 2018 (cited on page 1).

[7] MITRE Corporation. 'Mitre att&ck framework', Accessed: 2023. [Online]. Available: https://attack.mitre.org/ (cited on page 2).

[8] D. H. Jonassen, *Constructivist Learning Environments*. Educational Technology Publications, 1999 (cited on page 2).

[9] M. Bishop, 'Education in information security', *IEEE Security & Privacy*, vol. 1, no. 1, pp. 56–59, 2000 (cited on page 2).

[10] S. A. Triantafyllou, 'Constructivist Learning Environments', *5th International Conference on Advanced Research in Teaching an Educationa*, 2022 (cited on page 2).

[11] O. K. Akram, 'DEVELOPMENT OF A FRAMEWORK FOR HERITAGE CONSERVATION MANAGEMENT OF ARAB DISTRICT , ERBIL , IRAQ', Doctoral Thesis, Universiti Putra Malaysia, Putra, 2020 (cited on pages 3, 39, 40).

[12] OWASP Foundation. 'OWASP Top 10 – 2021', Accessed: 5th May 2025. [Online]. Available: https://owasp.org/www-project-top-ten/ (cited on page 9).

[13] M. Babiker, 'Web Application Attack Detection and Forensics : A Survey', *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, 2018 (cited on page 10).

[14] A. S. Hussainy, M. A. Khalifa, A. Elsayed, A. Hussien and M. A. Razek, 'Deep Learning Toward Preventing Web Attacks', *2022 5th International Conference on Computing and Informatics (ICCI)*, pp. 280–285, 2022. DOI: 10.1109/ICCI54321.2022.9756057 (cited on page 10).

[15] X. Li, M. Gao, W. Rong, Q. Xiong and J. Wen, 'Shilling Attacks Analysis in Collaborative Filtering Based Web Service Recommendation Systems', *2016 IEEE International Conference on Web Services*, 2016. DOI: 10.1109/ICWS.2016.75 (cited on page 11).

[16] V. Clincy and H. Shahriar, 'Web service injection attack detection', *2018 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, pp. 173–178, 2018. DOI: 10.23919/ICITST.2017.8356371 (cited on page 11).

[17] A. D. Khairkar, D. D. Kshirsagar and S. Kumar, 'Ontology for detection of web attacks', *2013 International Conference on Communication Systems and Network Technologies (CSNT)*, pp. 612–615, 2013. DOI: 10.1109/CSNT.2013.131 (cited on page 12).

[18] O. Ezenwoye and Y. Liu, 'Web Application Weakness Ontology Based on Vulnerability Data', *Augusta University*, pp. 1–16, 2020. arXiv: arXiv:2209.08067v1 (cited on page 12).

[19] A. A. Almutairi, S. Mishra and M. Alshehri, 'Web Security : Emerging Threats and Defense', *Computer Systems Science & Engineering*, vol. 40, no. 3, 2022. DOI: 10.32604/csse.2022.019427 (cited on page 13).

[20] P. A. Sonewar and S. D. Thosar, 'Detection of SQL injection and XSS attacks in three tier web applications', *2016 2nd International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, 2017. DOI: 10.1109/ICCUBEA.2016.7860069 (cited on page 13).

[21] G. P. Bherde and M. A. Pund, 'Recent attack prevention techniques in web service applications', *2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT)*, pp. 1174–1180, 2017. DOI: 10.1109/ICACDOT.2016.7877771 (cited on page 13).

[22] P. Bisht, M. S. Rauthan and R. K. Bisht, 'Component Based Web Application Firewall for Analyzing and Defending SQL Injection Attack Vectors', *International Journal of Recent Technology and Engineering*, 2019. DOI: 10.35940/ijrte.C4674.098319 (cited on pages 16, 47).

[23] D. Kapil, V. Mittal and D. P. Gangodkar, 'Virtualization and Nested Virtualization Technology: Concept, Architecture and Attack Vector Model', *2022 International Conference on Computational Intelligence and Sustainable Engineering Solution (CISES)*, pp. 349–354, 2022. DOI: 10.1109/CISES54857.2022.9844347 (cited on page 16).

[24] N. Mishra and R. K. Singh, 'Taxonomy Analysis of Cloud Computing Vulnerabilities through Attack Vector, CVSS and Complexity Parameter', *2019 2nd International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*, 2019. DOI: 10.1109/ICICT46931.2019.8977667 (cited on pages 17, 31, 38).

[25] P. L. Gallegos-Segovia, P. E. Vintimilla-Tapia, J. F. Bravo-Torres, I. F. Yuquilima-Albarado, V. M. Larios-Rosillo and J. D. Jara-Saltos, 'Social engineering as an attack vector for ransomware', *2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, pp. 1–6, 2017. DOI: 10.1109/CHILECON.2017.8229528 (cited on page 17).

[26] L. Ilsenstein, M. Koch and H. Steinhart, 'Definition of attack vectors to detect possible cyber-attacks on electrical machines', *2020 PCIM Europe Conference Proceedings*, vol. 1, pp. 1150–1156, 2020, ISSN: 21913358 (cited on page 18).

[27] M. Noyes and H. Alamleh, 'Input Fuzzing for Network-based Attack Vector on Smartphones', *2022 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, 2022. DOI: 10.1109/IEMTRONICS55184.2022.9795846 (cited on page 18).

[28] V. Vasilyev, A. Kirillova, A. Vulfin and A. Nikonov, 'Cybersecurity Risk Assessment Based on Cognitive Attack Vector Modeling with CVSS Score', *2021 - 7th IEEE International Conference on Information Technology and Nanotechnology*, 2021. DOI: 10.1109/ITNT52450.2021.9649191 (cited on page 19).

[29] F. Antonucci and M. M. Chowdhury, 'Botnets as the Modern Attack Vector', *2022 IEEE World AI IoT Congress, AIIoT*, pp. 585–590, 2022. DOI: 10.1109/AIIoT54504.2022.9817360 (cited on page 19).

[30] V. Susukailo, I. Opirskyy and S. Vasylyshyn, 'Analysis of the attack vectors used by threat actors during the pandemic', *2020 15th International Conference on Computer Sciences and Information Technologies*, vol. 2, pp. 261–264, 2020. DOI: 10.1109/CSIT49958.2020.9321897 (cited on pages 20, 50, 51).

[31] T. Mason and B. Zhou, 'Digital Forensics Process of an Attack Vector in ICS environment', *2021 IEEE International Conference on Big Data (Big Data)*, pp. 2532–2541, 2021. DOI: 10.1109/BigData52589.2021.9671986 (cited on page 20).

[32] J. N. Brewer and G. DImitoglou, 'Evaluation of attack vectors and risks in automobiles and road infrastructure', *2019 International Conference on Computational Science and*

Computational Intelligence (CSCI), pp. 84–89, 2019. DOI: 10.1109/CSCI49370.2019.00021 (cited on page 22).

[33] Y. Zhang et al., 'Looking Back! Using Early Versions of Android Apps as Attack Vectors', *2021 Transactions on Dependable and Secure Computing*, vol. 18, no. 2, pp. 652–666, 2021, ISSN: 19410018. DOI: 10.1109/TDSC.2019.2914202 (cited on page 23).

[34] E. Arshad, G. Turri and B. Crispo, 'Mollywood: Subtitles as an attack vector', *2022 Symposium on Computers and Communications (ISCC)*, 2022, ISSN: 15301346. DOI: 10.1109/ISCC55528.2022.9912810 (cited on page 24).

[35] M. Yin, Q. Wang and M. Cao, 'An Attack Vector Evaluation Method for Smart City Security Protection', *2019 The Second International Workshop on Big Data and Edge Computing for Smart City*, 2019, ISSN: 21619654. DOI: 10.1109/WiMOB.2019.8923576 (cited on page 25).

[36] X. Guo, S. Jin and Y. Zhang, 'XSS Vulnerability Detection Using Optimized Attack Vector Repertory', *2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pp. 29–36, 2015. DOI: 10.1109/CyberC.2015.50 (cited on page 26).

[37] D. E. Simos, J. Zivanovic and M. Leithner, 'Automated combinatorial testing for detecting SQL vulnerabilities in web applications', *2019 IEEE/ACM 14th International Workshop on Automation of Software Test (AST)*, pp. 55–61, 2019. DOI: 10.1109/AST.2019.00014 (cited on pages 26, 27).

[38] G. Kaur, B. Pande, A. Bhardwaj, G. Bhagat and S. Gupta, 'Efficient yet Robust Elimination of XSS Attack Vectors from HTML5 Web Applications Hosted on OSN-Based Cloud Platforms', *6th International Conference on Smart Computing and Communications (ICSCC)*, vol. 125, pp. 669–675, 2017, ISSN: 1877-0509. DOI: 10.1016/j.procs.2017.12.086 (cited on pages 28, 29).

[39] D. Bhatnagar, S. Som and S. K. Khatri, 'Advance Persistant Threat and Cyber Spying - The Big Picture, Its Tools, Attack Vectors and Countermeasures', *2019 Amity*

International Conference on Artificial Intelligence (AICAI), pp. 828–839, 2019. DOI: 10.1109/AICAI.2019.8701329 (cited on page 30).

[40] M. J. Haber, *Privileged Attack Vectors*, 2nd ed., S. McDermott, L. Berendson and R. Fernando, Eds. Heathrow, FL, USA: Apress, 2020, ch. 4, pp. 66–85, ISBN: 9781484259139 (cited on pages 30–32, 38, 47).

[41] R. Derbyshire, B. Green, D. Prince, A. Mauthe and D. Hutchison, 'An Analysis of Cyber Security Attack Taxonomies', *2018 European Symposium on Security and Privacy Workshops*, pp. 153–161, 2018. DOI: 10.1109/EuroSPW.2018.00028 (cited on page 33).

[42] C. Easttom and W. Butler, 'A Modified McCumber Cube as a Basis for a Taxonomy of Cyber Attacks', *2019 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 943–949, 2019. DOI: 10.1109/CCWC.2019.8666559 (cited on pages 34, 38).

[43] J. Komárková, L. Sadlek and M. Laštovička, 'Community Based Platform for Vulnerability Categorization', *2018 IEEE/IFIP Network Operations and Management Symposium: Cognitive Management in a Cyber World (NOMS)*, pp. 1–2, 2018. DOI: 10.1109/NOMS.2018.8406125 (cited on page 35).

[44] J. Y. Lai, J. S. Wu, S. J. Chen, C. H. Wu and C. H. Yang, 'Designing a Taxonomy of Web Attacks', *2008 International Conference on Convergence and Hybrid Information Technology (ICHIT)*, pp. 278–282, 2008. DOI: 10.1109/ICHIT.2008.280 (cited on pages 35, 36).

[45] R. K. Yin, *Case Study Research: Design and Methods*, 3rd ed., WordPress, Ed. Thousand Oaks, London and Nova Delhi: Sage Publications, Inc., 2009, vol. 5, pp. 1–318 (cited on pages 37, 38).

[46] OWASP Foundation. 'Owasp top 10 project repository', Accessed: 26th Mar. 2026. [Online]. Available: https://github.com/OWASP/Top10 (cited on pages 44–46).

[47] Forum of Incident Response and Security Teams, Inc. All. 'Common vulnerability scoring system version 4.0 specification document', Accessed: 2025. [Online]. Available: https://www.first.org/cvss/ (cited on pages 46, 53–59, 61–64, 67).

[48] A. Malik, S. Gehlot and A. Aggarwal, 'Attacks on web applications', *Perspectives on Ethical Hacking and Penetration Testing*, pp. 31–62, Sep. 2023. DOI: 10.4018/978-1-6684-8218-6.ch002 (cited on page 47).

[49] Fortinet Training. 'Fcf - introduction to the threat landscape 3.0', Accessed: 2025. [Online]. Available: https://training.fortinet.com/ (cited on pages 47, 51–53).

[50] Fortinet Training. 'Attack vector', Accessed: 26th May 2025. [Online]. Available: https://www.fortinet.com/resources/cyberglossary/attack-vector (cited on page 48).

[51] National Institute of Standards and Technology (NIST). 'Guide for conducting risk assessments (sp 800-30 rev. 1)', Accessed: 5th Oct. 2023. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-30/rev-1/final (cited on pages 48, 49).

[52] OWASP Foundation. 'Owasp web security testing guide', Accessed: 5th Oct. 2023. [Online]. Available: https://owasp.org/www-project-web-security-testing-guide/ (cited on page 48).

[53] MITRE Corporation. 'Common weakness enumeration (cwe)', Accessed: 5th Oct. 2023. [Online]. Available: https://cwe.mitre.org/ (cited on page 48).

[54] R. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, 3rd ed. Wiley, 2020, ch. 2 and 3, pp. 17–118 (cited on pages 48–50).

[55] D. Stuttard and M. Pinto, *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws*, 2nd ed. Wiley, 2011 (cited on pages 48, 49, 82).

[56] Verizon. 'Data breach investigations report 2023', Accessed: 2025. [Online]. Available: https://www.verizon.com/business/resources/reports/dbir/ (cited on pages 49, 50).

[57] European Union Agency for Cybersecurity (ENISA). 'Enisa threat landscape report: Cybersecurity trends and threats', Accessed: 2025. [Online]. Available: https://www.enisa.europa.eu/publications (cited on pages 49, 50).

[58] Centraleyes. 'Cyber risk management platform', Accessed: 2025. [Online]. Available: https://www.centraleyes.com/ (cited on page 54).

[59] A. Karasev. 'The transition to cvss v4.0 – what you need to know', Accessed: 2025. [Online]. Available: https://tuxcare.com/blog/cvss-v4-0/ (cited on page 58).

[60] M. Teixeira. 'Cvss v4.0: What you need to know about the latest version', Accessed: 2025. [Online]. Available: https://checkmarx.com/blog/cvss4-what-you-need-to-know-about-the-latest-version/ (cited on page 58).

[61] Forum of Incident Response and Security Teams, Inc. All (first.org). 'Cvss v4.0 frequently asked questions', Accessed: 2025. [Online]. Available: https://www.first.org/cvss/v4-0/faq (cited on page 58).

[62] FIRST Foundation. 'Cvss v4.0 examples', Accessed: Jul. 2025. [Online]. Available: https://www.first.org/cvss/v4-0/examples (cited on pages 58–62, 65–69).

[63] S. Deterding, D. Dixon, R. Khaled and L. Nacke, 'From game design elements to gamefulness: Defining "gamification"', in *MindTrek 2011*, 2011, pp. 9–15 (cited on pages 72, 73, 86).

[64] A. Domínguez, J. Saenz-de-Navarrete et al., 'Gamifying learning experiences: Practical implications and outcomes', *Computers & Education*, vol. 63, pp. 380–392, 2013 (cited on pages 72, 73, 86).

[65] A. Sasse and I. Flechais, 'Usable security: Why do we need it? how do we get it?', in *Security and Usability*, O'Reilly, 2005, ch. 2, pp. 13–30 (cited on pages 72, 73).

[66] P. Puhakainen and M. Siponen, 'Design theory for information security awareness', *Journal of the Association for Information Systems*, vol. 11, no. 7, pp. 519–555, 2010 (cited on pages 72, 73).

[67] National Institute of Standards and Technology, 'Framework for improving critical infrastructure cybersecurity', NIST, Tech. Rep., 2018 (cited on pages 72, 73).

[68] Y. Acar, C. Stransky, D. Wermke, C. Weir, M. L. Mazurek and S. Fahl, 'Developers need support, too: A survey of security advice for software developers', *2017 Cybersecurity Development Conference (SecDev)*, pp. 22–26, 2017. DOI: 10.1109/SecDev.2017.17 (cited on pages 72, 73, 86).

[69] S. Fahl, M. Harbach, T. Muders and M. Smith, 'Why eve and mallory love android: An analysis of android ssl (in)security', *2012 ACM Conference on Computer and Communications Security*, pp. 50–61, 2012 (cited on pages 72, 73).

[70] J. Piaget and BarbelInhelder, *The Psychology of the Child*. 10 East 53rd Street, New York: Basic Books, Inc., 2000 (cited on pages 72, 85).

[71] L. S. Vygotsky, *Mind in Society: The Development of Higher Psychological Processes*. Harvard University Press, 1978, ISBN: 0–674-57629-2 (cited on pages 72, 85).

[72] J. Bruner, *The Culture of Education*. Harvard University Press, 1996, ch. 2 (cited on page 72).

[73] D. A. Kolb, *Experiential Learning: Experience as the Source of Learning and Development*. Englewood Cliffs, NJ: Prentice-Hall, 1984, ISBN: 9780132952613 (cited on pages 72, 85, 92).

[74] R. C. Schank, 'What we learn when we learn by doing', Institute for the Learning Sciences, Northwestern University, Evanston, IL, Technical Report 60, 1995 (cited on page 72).

[75] M. Prince, 'Does active learning work? a review of the research', *Journal of Engineering Education*, vol. 93, no. 3, pp. 223–231, 2004. DOI: 10.1002/j.2168-9830.2004.tb00809.x (cited on page 72).

[76] J. D. Bransford, A. L. Brown and R. R. Cocking, *How People Learn: Brain, Mind, Experience, and School*. Washington, D.C.: National Academy Press, 2000, ISBN: 978-0-309-07036-2 (cited on page 72).

[77] T. Chothia, S. Holdcroft, A.-I. Radu and R. J. Thomas, 'Jail, hero or drug lord? turning a cyber security course into an 11 week choose your own adventure story', *USENIX Workshop on Advances in Security Education (ASE)*, 2017 (cited on pages 73, 77, 86, 87).

[78] National Initiative for Cybersecurity Education (NICE), 'The cyber range: A guide', National Institute of Standards and Technology, Gaithersburg, MD, Guidance Document, 2020 (cited on pages 73, 76, 77, 82, 85, 87).

[79] N. Chouliaras, G. Kambourakis, G. Karamitas and S. Gritzalis, 'Cyber ranges and testbeds for education, training, and research', *Applied Sciences*, vol. 11, no. 4, p. 1809, 2021. DOI: 10.3390/app11041809 (cited on page 73).

[80] V. Švábenský, P. Čeleda, J. Vykopal and S. Brišáková, 'Cybersecurity knowledge and skills taught in capture the flag challenges', *Elsevier Computers & Security*, vol. 19, no. 5, pp. 67–76, 2021. DOI: 10.1109/MSEC.2021.3095716 (cited on pages 74, 80).

[81] J. Venable, J. Pries-Heje and R. Baskerville, 'Feds: A framework for evaluation in design science research', *2016 European Journal of Information Systems*, vol. 25, no. 1, pp. 77–89, 2016. DOI: 10.1057/ejis.2014.36 (cited on page 74).

[82] K. Beck et al. 'Manifesto for agile software development', Accessed: 11th Nov. 2025. [Online]. Available: https://agilemanifesto.org/ (cited on page 75).

[83] S. John, N. Shah and L. Smalov, 'Incremental and iterative agile methodology (iiam):hybrid approach for ontology design towards semanticweb based educational systems development', *International Journal of Knowledge Engineering*, vol. 2, no. 1, pp. 13–19, 2016 (cited on page 75).

[84] A. R. Hevner and S. Chatterjee, *Design Research in Information Systems: Theory and Practice*. Claremont CA: Springer, 2010, ISBN: 978-1-4419-5653-8 (cited on page 75).

[85] P. Chen and M. Rosenblum, 'When virtual is better than real', *Department of Electrical Engineering and Computer Science University of Michigan*, 2021 (cited on page 76).

[86] T. Ristenpart, E. Tromer, H. Shacham and S. Savage, 'Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds', *2009 16th ACM Conference on Computer and Communications Security (CCS)*, pp. 199–212, 2009. DOI: 10.1145/1653662.1653687 (cited on page 76).

[87] D. Merkel, 'Docker: Lightweight linux containers for consistent development and deployment', *Linux Journal*, vol. 2014, no. 239, p. 2, 2014 (cited on page 76).

[88] M. Xavier, M. Neves, F. Rossi et al., 'Performance evaluation of container-based virtualization for high performance computing environments', *2013 21st Euromicro*

International Conference on Parallel, Distributed, and Network-Based Processing, 2013. DOI: 10.1109/PDP.2013.41 (cited on page 76).

[89] T. Benzel, 'The science of cyber security experimentation: The deter project', *2011 Annual Computer Security Applications Conference (ACSAC)*, vol. 9, no. 2, 76–79 and 137–148, 2011 (cited on pages 76, 82, 83, 85, 87).

[90] R. Sommer and V. Paxson, 'Outside the closed world: On using machine learning for network intrusion detection', in *2010 Symposium on Security and Privacy*, IEEE, 2010, pp. 305–316 (cited on page 76).

[91] E. Ukwandu, M. A. B. Farah, H. Hindy et al., 'A review of cyber-ranges and test-beds: Current and future trends', *Sensors*, vol. 96, pp. 101–115, 2020 (cited on pages 76, 82, 83).

[92] P. Ammann and J. Offutt, *Introduction to Software Testing*. Cambridge CB2 8RU, UK: Cambridge University Press, 2016, pp. 1–10, ISBN: 978-0-521-88038-1 (cited on pages 77, 87).

[93] B. Meyer, 'On formalism in specifications', *2009 International Conference on Software Engineering*, pp. 1–10, 2009 (cited on page 77).

[94] V. Švábenský, R. Weiss, J. Cook, J. Vykopal and P. Čeleda, 'Evaluating two approaches to assessing student progress in cybersecurity exercises', *2021 53rd ACM Technical Symposium on Computer Science Education*, vol. 1, pp. 787–793, 2021 (cited on pages 77, 82).

[95] M. Karjalainen and T. Kokkonen, 'Comprehensive cyber arena: The next generation cyber range', *2020 European Symposium on Security and Privacy Workshops (EuroS&PW)*, pp. 11–16, 2020. DOI: 10.1109/EuroS&PW51379.2020.00010 (cited on page 77).

[96] M. N. Katsantonis, A. Manikas, I. Mavridis and D. Gritzalis, 'Cyber range design framework for cyber security education and training', *2023 International Journal of Information Security*, vol. 22, pp. 1005–1027, 2023. DOI: 10.1007/s10207-023-00680-4 (cited on pages 77, 82, 83).

[97] B. Kimminich, *The OWASP Juice Shop Project*. OWASP Foundation, 2018. Accessed: 2025. [Online]. Available: https://owasp.org/www-project-juice-shop/ (cited on page 79).

[98] Damn Vulnerable Web Application Project. 'Damn vulnerable web application (dvwa)', Accessed: 2025. [Online]. Available: https://github.com/digininja/DVWA (cited on page 79).

[99] OWASP Foundation. 'Owasp webgoat project', Accessed: 2025. [Online]. Available: https://owasp.org/www-project-webgoat/ (cited on page 79).

[100] Hack The Box. 'Hack the box platform', Accessed: 2025. [Online]. Available: https://www.hackthebox.com (cited on page 80).

[101] TryHackMe. 'Tryhackme: Cyber security training', Accessed: 2025. [Online]. Available: https://tryhackme.com (cited on page 80).

[102] PortSwigger Ltd. 'Web security academy', Accessed: 2025. [Online]. Available: https://portswigger.net/web-security (cited on page 80).

[103] K. Chung. 'Ctfd: Capture the flag framework', Accessed: 2025. [Online]. Available: https://github.com/CTFd/CTFd (cited on page 80).

[104] RootTheBox Project. 'Roottheboxx: A gamified hacking platform', Accessed: 2025. [Online]. Available: https://github.com/moloch--/RootTheBox (cited on page 80).

[105] J. H. Saltzer and M. F. Kaashoek, *Principles of Computer System Design: An Introduction*. Morgan Kaufmann, 2009 (cited on page 83).

[106] S. Newman, *Building Microservices*. Sebastopol, CA: O'Reilly Media, 2015, ISBN: 9781491950357 (cited on page 83).

[107] D. Hardt, 'The oauth 2.0 authorization framework', *Internet Engineering Task Force (IETF)*, 2012. DOI: 10.17487/RFC6749 (cited on page 83).

[108] M. Lathkar, *Building Web Apps with Python and Flask*, 1st ed. BPB Publications, 2021 (cited on pages 84, 92–100).

[109] J. H. Saltzer and M. D. Schroeder, 'The protection of information in computer systems', *Information Systems*, 1975 (cited on pages 85, 88, 113).

[110] A. Shostack, *Threat Modeling: Designing for Security*. Indianapolis, IN: John Wiley & Sons, 2014, ISBN: 978-1118809990 (cited on pages 85, 113).

[111] J. Vykopal, R. Ošlejšek, P. Čeleda, M. Vizváry and D. Tovarnák, 'Kypo cyber range: Design and use cases', *12th International Conference on Software Technologies (ICSOFT)*, pp. 310–321, 2017. DOI: 10.5220/0006428203100321 (cited on pages 87, 88, 101).

[112] W. Newhouse, S. Keith, B. Scribner and G. Witte, 'National initiative for cybersecurity education (nice) cybersecurity workforce framework', National Institute of Standards and Technology, Tech. Rep. 800-181, 2020. DOI: 10.6028/NIST.SP.800-181r1 (cited on page 88).

[113] P. J. Eby. 'Pep 3333 – python web server gateway interface v1.0.1', Accessed: 2025. [Online]. Available: https://peps.python.org/pep-3333/ (cited on page 92).

[114] D. Lord. 'Flask documentation', Accessed: 26th May 2025. [Online]. Available: https://flask.palletsprojects.com (cited on pages 92, 94).

[115] L. Ramalho, *Fluent Python, Clear, Concise, and Effective Programming*, 2nd Edition. O'Reilly Media Inc., 2022 (cited on page 94).

[116] D. S. Foundation. 'Django project documentation', Accessed: 26th May 2025. [Online]. Available: https://www.djangoproject.com/ (cited on page 94).

[117] S. Tiangolo. 'Fastapi documentation', Accessed: 26th May 2025. [Online]. Available: https://fastapi.tiangolo.com/ (cited on page 94).

[118] Apache Software Foundation. 'Apache http server documentation: Cgi', Accessed: Jul. 2025. [Online]. Available: https://www.apache.org/docs/ (cited on page 95).

[119] R. T. Fielding, 'Architectural Styles and the Design of Network-based Software Architectures', Ph.D. dissertation, University of California, Irvine, 2000 (cited on page 96).

[120] S. Arragokula and M. Y. Ratnam, 'Architectural Styles and the Design of Network-based Software Architectures ', *International Journal of Ethics in Engineering & Management Education*, vol. 3, 2016 (cited on page 96).

[121] Gunicorn Project. 'Gunicorn documentation', Accessed: Jan. 2026. [Online]. Available: https://gunicorn.org/ (cited on page 97).

[122] L. Bass, P. Clements and R. Kazman, *Software Architecture in Practice*, 3rd ed. Boston, MA: Addison-Wesley, 2012, ISBN: 9780321815736 (cited on page 101).

[123] D. Merkel, 'Docker: Lightweight linux containers for consistent development and deployment', *Linux Journal*, vol. 2014, no. 239, p. 2, 2014 (cited on page 101).

[124] OWASP Foundation. 'Owasp threat modeling', Accessed: 2026. [Online]. Available: https://owasp.org/www-community/Threat_Modeling (cited on page 113).

[125] M. Howard and D. LeBlanc, *Writing Secure Code*, 2nd ed. Washington, DC: Microsoft Press, 2006, ISBN: 0-7356-1722-8 (cited on page 113).

Glossaries

.NET	is a free, open-source software development platform developed by Microsoft. It allows developers to build a wide range of applications, including web apps, desktop apps, and mobile apps, using various programming languages like C#, F#, and VB.NET. .NET runs natively on multiple operating systems, including Windows, Linux, and macOS, and also supported on Android and Apple devices. 11
API	stands for Application Programming Interface. It's a set of rules and protocols that allows different software applications to communicate and exchange data with each other. In essence, an API defines how applications interact, enabling them to request and receive data or functionality from other systems. 15
Bot	in cybersecurity, it is a software application programmed to perform repetitive tasks automatically. 52
botnet	is a network of computers or devices that have been infected with malware and are under the control of a hacker or cybercriminal. xix, 19, 52, 53
Burp Suite	is a powerful web application security testing tool developed by PortSwigger. 26

CloudFlare	is a cloud-based service that provides a range of network services, including content delivery networks (CDNs), web security, and Domain Name System (DNS) services. 19
CPU	or Central Processing Unit, is the brain of a computer, responsible for executing instructions and performing calculations. 24
CSS	Cyber Security Standard is an independent and standalone framework certification tailored specifically to the needs and requirements of the supply chain industry. 26
Domain Name System (DNS)	acts like a telephone book for the internet, translating human-readable domain names (like www.example.com) into numerical IP addresses that computers use to communicate. Without DNS, users would have to navigate the web by entering IP addresses, which are difficult to remember and prone to errors. 60, 150
ESS	Ease of Exploitation Score, a metric used in vulnerability scoring systems to represent how easily a vulnerability can be exploited. 65
ESXi	formerly ESX, is an enterprise-class, type-1 hypervisor developed by VMware, a subsidiary of Broadcom, for deploying and serving virtual computers. 16

GPU	is a specialised processor designed to handle complex mathematical calculations, especially those related to rendering images and videos. 24
honeynet	is a decoy system or network designed to attract and trap cyber attackers. 19
honeypot	is a decoy system or network designed to attract and trap cyber attackers. 9, 19
Hyper-v	is a Microsoft virtualisation technology that allows users to create and run virtual machines (VMs) on a single physical server or computer. 16
IBM	stands for International Business Machines Corporation. It is a multinational technology company known for its hardware, software, and services. 16
Java	in programming, it refers to a widely used, high-level, object-oriented programming language, not an abbreviation with a full form. While some jokingly say ”Java” stands for ”Just Another Virtual Accelerator”, the Indonesian coffee Java inspires it. James Gosling and his team at Sun Microsystems created the language in 1995. 12
KVM	is an open-source virtualisation technology for Linux that allows running multiple virtual machines (VMs) on a single physical machine. 16

MITM	Man-in-the-Middle is a type of cyberattack where an attacker intercepts and relays messages between two parties who believe they are communicating directly with each other. 23
payload	in cybersecurity, a payload refers to the malicious part of malware or an exploit that causes harm to a system or network. 49
PHP	initially, it was an acronym for Personal Home Page Tools, but it now refers to itself, making it a recursive acronym. PHP is a widely used, open-source scripting language primarily for web development, particularly server-side scripting. 12, 79
SPAM	refers to unsolicited and unwanted messages, often sent in bulk, via various digital communication channels like email, social media, or messaging apps. 52
sqlmap	is an open-source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of back-end database servers. 26
STRIDE	refers to a threat modeling framework developed by ®Misrosoft to identify and classify security risks during the design phase of software development 113, 114, 120

SYN	typically refers to ”synchronise,” and it’s used in the context of TCP (Transmission Control Protocol) connections. 20
URL	is a unique web address that identifies a specific resource on the internet, such as a website, file, or image. 26, 28
Virtual PC	is a discontinued x86 emulator software-based, or virtual, version of a computer that runs on top of a physical host computer for Microsoft Windows hosts. 16
VirtualBox	is a type of virtualisation software, from Oracle®, that allows users to run multiple operating systems on a single computer. 16
VMware	is a company that develops virtualisation and cloud computing software, allowing multiple virtual machines (VMs) to run on a single physical server. 16
Web 2.0	Web 2.0 transforms the World Wide Web from static, one-way communication into a more dynamic, interactive, and social experience. 17, 49
World Wide Web	is an information system on the internet which allows documents to be connected to other documents by hypertext links, enabling the user to search for information by moving from one document to another. 12, 95
WPA	is an attack vector model based on weakness, path, and action. 24, 25

Xen	is a type of hypervisor that allows multiple operating systems to run on the same physical hardware concurrently. 16
XML	is a metalanguage which allows users to define their customised markup languages, especially to display documents on the internet. Metalanguage enables users to define their custom markup languages, particularly for displaying documents on the internet. 15

Appendices

Appendix A

Direcotry Structure

Figure A.1

Directory structure of the Portal VM (VM1).

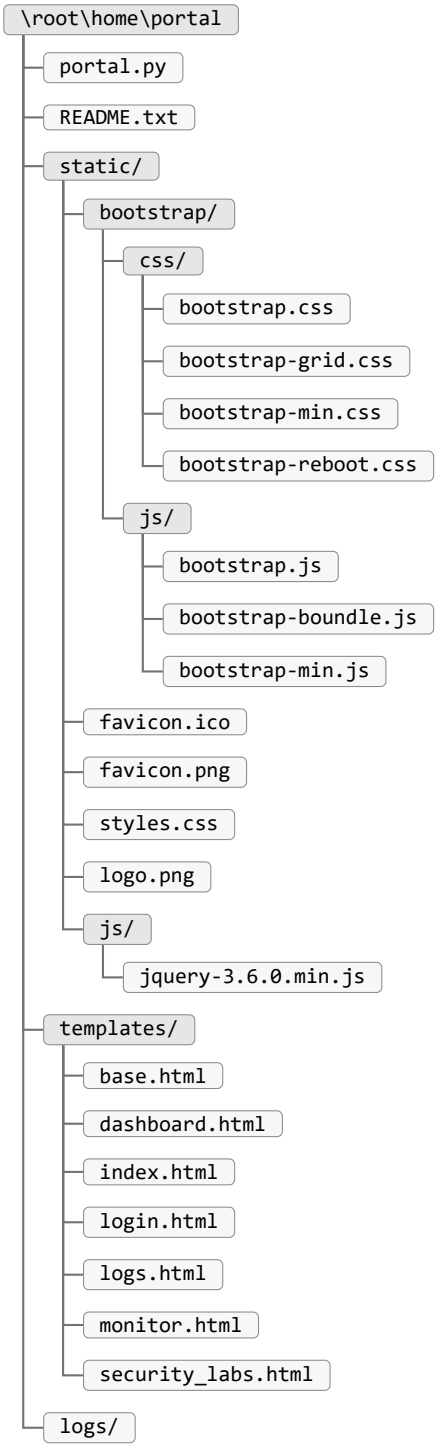
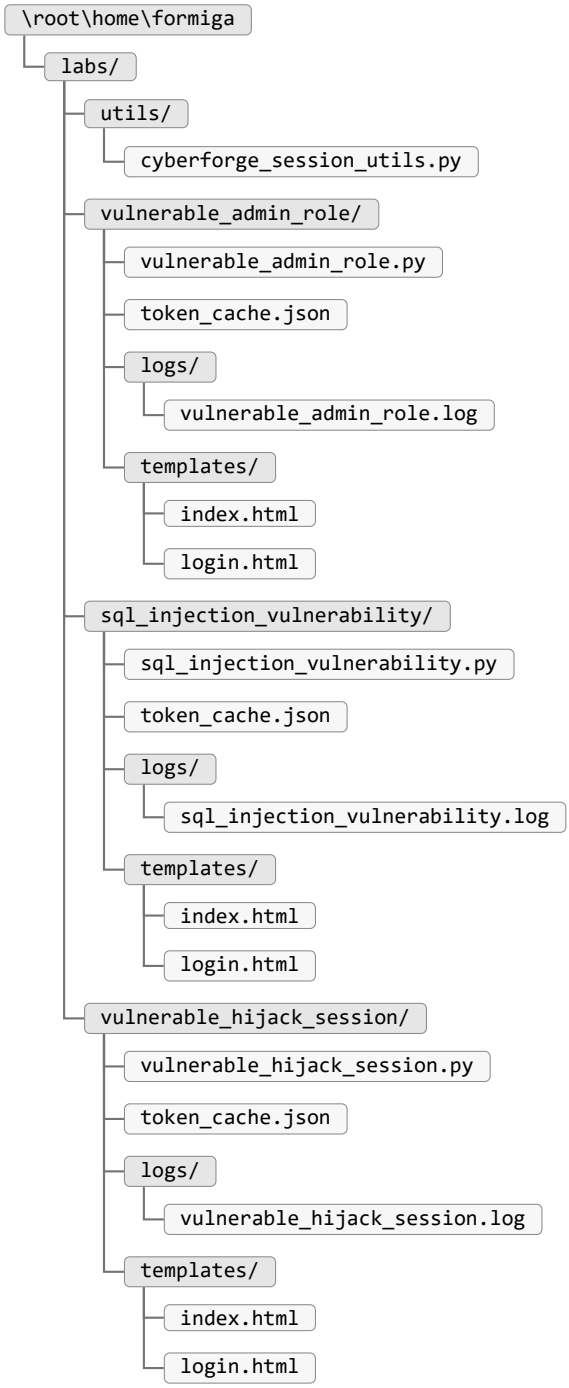


Figure A.2

Directory structure of the Scenarios VM (VM2).



Assinado por: **João Carlos Modesto Parra**
Num. de Identificação: 06938878
Data: 2026.07.01 19:10:52+01'00'

CHAVE MÓVEL