AN AI-POWERED NETWORK THREAT ASSESSMENT AND RECOMMENDATION SYSTEM

BY

MICHEAL IFEANYICHUKWU MKPARU

GOU/U22/CSC/783

DEPARTMENT OF COMPUTER SCIENCE AND MATHEMATICS, FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY,

GODFREY OKOYE UNIVERSITY, ENUGU STATE. IN PARTIAL FULFILLMENT OF THE AWARD OF BACHELOR OF

SCIENCE (B.SC), DEGREE IN COMPUTER SCIENCE.

SUPERVISOR: PROF. IFEYINWA A AJAH

JUNE, 2026.

APPROVAL PAGE

This research, titled "**AI-POWERED NETWORK THREAT ASSESSMENT AND RECCOMENDATION SYSTEM**," has been assessed and approved by the Department of Computer Science and Mathematics Committees in the Faculty of computing and information technology, Godfrey Okoye University, Enugu, Nigeria.

Prof. Ifeyinwa A Ajah Supervisor:	_____ Signature	_____ Date
External Examiner External Examiner	_____ Signature	_____ Date
Dr. Frank Okebanama HOD, Department of Computer Science and Mathematics	_____ Signature	_____ Date
Prof. Ifeyinwa A Ajah Dean, Faculty of Computing And Information Technology.	_____ Signature	_____ Date

CERTIFICATION

I certify that I completed the research included therein and that it was primarily based on my observations and investigation. Consequently, I will fully accept the University's decision if I am found to have cheated on the assignment.

_____

MICHEAL IFEANYICHUKWU MKPARU

GOU/U22/CSC/783

DEDICATION

I dedicate this project to God for His guidance and protection throughout my journey in this university, and also to my family and friends, for their endless support and encouragement which has heavily contributed to the success of this project.

ACKNOWLEDGMENTS

My profound gratitude to God, who has guided and given me the knowledge and inspiration to embark on this project.

My sincere appreciation goes to my project supervisor Prof. Ifeyinwa A Ajah who provided me with valuable advices, ideas and guidance throughout the course of this project. Her unwavering support and expertise were instrumental in ensuring this project was a success.

To my parents and guardians for their support, prayers and financial provision, to my friends Pecu, Kene, Chris, Elo, Sophie for their care, love and support and to my colleagues who contributed one way or the other to the success of this project and making the journey a less-stressful and blissful one, I am sincerely grateful.

ABSTRACT

The rise of digitalization in academic institutions has increased the vulnerability of campus networks to cybercriminal activities. Existing approaches to vulnerability scanning have been found to be dependent on the Common Vulnerability Scoring System (CVSS). The problem with CVSS is that it has been found to provide poor estimates regarding exploitability of vulnerabilities, leading to inefficiencies in security resource allocation. This has led to alert fatigue and late remediation of threats, especially in limited-resource settings such as Nigerian universities. The present paper concentrated on the design and development of an AI-Powered Network Threat Assessment and Recommendation System to prioritize, analyze, and interpret the vulnerability in the university network environment. The developed system uses an XGBoost binary classifier which has been trained on 155,852 real-world CVEs of a Kaggle CVE/EPSS/KEV enriched dataset using the CVSS score, criticality of assets, exploit likelihood (EPSS), and vulnerability age as predictors. The NLP has been used to come up with simplified, understandable vulnerability remediation suggestions. At the same time, the Explainable AI (XAI) approach using SHAP (Shapley Additive Explanation) makes the model's predictions transparent. The system harvests the vulnerability information from Nmap network scan, which is enriched with the help of live threat intelligence collected from the National Vulnerability Database (NVD), FIRST.org EPSS API, and CISA Known Exploited Vulnerabilities (KEV) database. Finally, a web-based dashboard has been developed in the Flask, Bootstrap, and Chart.js technologies. After the training process, the obtained AUC-ROC of the trained classifier on the holdout set was 0.946 (5-fold cross-validation AUC of 0.952 ± 0.009) and the recall was 80.6%. The system was able to get 90% Precision@50, meaning that 90% of the 50 highest-priority AI-ranks were confirmed exploited vulnerabilities, whereas for the CVSS-only ranking it was just 22%.

TABL OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

1.0 Background to Study

As times have evolved, the growth of digitized operations in universities has evolved too. The development of digital infrastructures has increased amongst Nigerian universities, and it has increased their reliance on network systems -online portals, e-learning systems, Wi-Fi networks, research databases, among others. Organizations like hospitals, education systems, enterprises, and even aviation systems are increasingly being targeted by hackers who exploit vulnerabilities in networks, software and system configurations. Public reports suggest that cyber-attacks have become more frequent, more professional and more expensive, with vulnerabilities being the major way hackers gain access (Verizon, 2024; ENISA, 2023).

Vulnerability refers to a weakness or flaw within the system which hackers exploit to affect the integrity of a system. Most organizations have traditionally used tools such as OpenVAS, Nessus, and Qualys to scan for vulnerabilities in their systems periodically. They produce an unorganized, long list of the vulnerabilities found in the system and they are prioritized according to the standard CVSS metric.

Although these tools are widely adopted, they often overwhelm security teams with large volumes of alerts and provide limited assistance on which vulnerabilities pose the most significant real-world risk (Allodi & Massacci, 2014; Holm et al., 2015).

Several studies have shown that CVSS-based severity scores alone are insufficient for effective vulnerability prioritization. In several studies, Shimizu and Hashimoto (2025) suggested that many times vulnerabilities that have high CVSS are hardly exploited in real time; instead, lower scoring vulnerabilities are often targeted. Shimizu and Hashimoto (2025) show that traditional

CVSS-centric prioritization flagged only about 20% of vulnerabilities actually exploited in real data, indicating that relying on CVSS alone has poor prioritization performance. This mismatch between theoretical severity and real-world exploitability results in inefficient allocation of security resources and delayed repair of critical threats.

Current studies have even more evidence that the conventional severity metrics like CVSS do not reflect the actual attack dynamics leading to the adoption of risk-based along with context-aware vulnerability assessment models taking exploit intelligence and context into account (Ponta et al., 2020).

Additionally, most traditional scanners do not have contextual awareness; they do not consider organizational environment, consider asset criticality, network exposure, or attack behavior when assessing risk.

To address these shortcomings, researchers have continuously explored the use of Artificial Intelligence (AI) and Machine Learning(ML) in cybersecurity. By the method of learning patterns from historical data, ML algorithms have been implemented to solve problems of intrusion detection, malware classification, and threat detection (Sommer & Paxson, 2010; Buczak & Guven, 2016). For example, Wang et al. (2022) proposed an AI-powered network threat-detection system that uses honeypot data to classify attack severity. While their approach achieved high detection accuracy, it was largely reactive, focusing on detecting attacks after they occurred rather than preventing them through proactive vulnerability management.

According to recent research, AI and ML are not only used for intrusion detection but also for vulnerability risk assessment and prioritization in security. This application of AI and ML allows security teams to concentrate on the vulnerabilities that are most likely to be exploited and have the greatest operational impact (Zhang et al., 2021; Almukaynizi et al., 2022).

Other studies have focused on predicting vulnerability exploitability. Allodi, Corradin, and Massacci (2021) proposed a data-driven approach that leverages real-world attack observations and vulnerability metadata to estimate the likelihood of exploitation, demonstrating that exploit prediction provides more actionable insights than static severity scores. Similarly, the Exploit Prediction Scoring System (EPSS), developed by the Forum of Incident Response and Security Teams (FIRST), estimates the probability that a vulnerability will be exploited within a given time frame based on empirical threat intelligence data (FIRST, 2022).

While these approaches represent significant improvements over traditional severity-based scoring systems, they still operate largely in isolation and do not provide comprehensive vulnerability management solutions that integrate organizational context, asset criticality, and remediation constraints.

While there are advances in this study, there are still gaps in existing research and tools. Most current tools or systems focus on threat detection or exploit prediction and use traditional security measures like firewalls and intrusion detection systems (IDSs), which are reactive and address attacks after they have happened. Additionally, a lot of AI-based security solutions are designed for large organizations and do not adequately address the needs of smaller environments that lack resources, like universities and small organizations, where skilled cybersecurity personnel may be limited.

This study aims to address these limitations. The AI-Powered Threat Assessment and Recommendation System is proposed to be an AI-powered smart vulnerability management system that prioritizes, analyzes, and explains vulnerabilities, making it easier for IT personnel to address and perform timely remediation.

1.1 Statement of Problems

Godfrey Okoye University, like many other universities, faces challenges in proactively identifying threats before they cause significant harm.

Institutions' reliance on network systems makes them targets for cybercriminals, which is why vulnerability scanners are used on these networks periodically to ensure software is up to date and that there are no other weaknesses in the system.

Traditional vulnerability scanning tools generate large volumes of vulnerability alerts but provide limited guidance on effective prioritization, overwhelming IT personnel and leading to alert fatigue.

Heavy dependence on CVSS-based severity scoring does not adequately reflect real-world exploitability or the contextual risk specific to environments like GOUNI.

Critical vulnerabilities affecting core university systems may remain unresolved, while less impactful issues receive disproportionate attention.

Most existing intelligent security tools are designed for large enterprise environments and do not adequately address the resource constraints of university networks.

There is a lack of an intelligent, explainable, and context-aware vulnerability management system tailored to support proactive cybersecurity decision-making within Godfrey Okoye University.

Many AI-based cybersecurity solutions are reactive, lack explainability, and are not seamlessly integrated into operational vulnerability management workflows.

Existing vulnerability management solutions lack contextual awareness, such as asset criticality, network exposure, and institutional risk factors, resulting in inefficient remediation efforts.

As a result of the problems outlined above, there is a need to develop an AI-powered threat assessment and recommendation system that prioritizes, analyzes, and explains vulnerabilities in a more user-friendly manner.

1.2 Aim and Objectives of the Study

This study aims to develop an AI-powered Network Threat Assessment and Recommendation System for easier and effective management of vulnerabilities in a campus network.

The objectives of the study are to:

1. Determine and assess weak points in a university network environment.
2. Build an ML-driven vulnerability ranking model that categorizes the vulnerabilities and intelligently decides which vulnerabilities should be fixed first based on how dangerous they are.
3. Create a ranked list of remediation advice that can be used to help system administrators deal with critical issues
4. Evaluate the effectiveness of the proposed system.

1.3 Significance of the study

The study is significant in the following ways:

1. It contributes to the field of cybersecurity
2. It is significant to university network admins and IT security personnel
3. It is significant to staff, students, and other users of university network systems, as improved vulnerability management systems mean strengthened protection of sensitive data
4. This research holds practical relevance for educational institutions in developing regions because the system is intended to be scalable, adaptable and user-friendly.

1.4 Scope of the study

This research is concentrated on the development of AI-Powered Threat Assessment and Recommendation System in order to handle vulnerabilities on university networks. The research is confined only to the process of assessment and management of the identified vulnerabilities on university networks. The proposed system uses Nmap for network asset discovery and a realworld CVE dataset sourced from Kaggle, containing 155,852 CVEs enriched with EPSS scores and CISA KEV flags as input for analysis. It applies AI and NLP techniques to evaluate and prioritize vulnerabilities and to generate remediation recommendations.

The study is within vulnerability management and threat assessment. It does not cover real-time intrusion detection, active attacks, or virus analysis. It is intended to be a proactive system that helps security/IT teams in decision-making.

The scope is limited to a simulated university network environment using VMware virtualization technology. Legal, regulatory, and policy compliance issues, as well as social engineering attacks and physical security threats, are outside the scope of this research.

1.5 Limitations of the study

The limitations in the study are:

First, the proposed system relies on vulnerability data obtained from Nmap scanning. Vulnerabilities not detectable through service version analysis may not be considered in the risk assessment process.

Secondly, while the ML model was trained on the Kaggle CVE/EPSS/KEV enriched dataset of 155,852 real CVE records, the quality and diversity of training data may still influence accuracy across different network environments.

Thirdly, the network assets used for evaluation were simulated using VMware, as direct access to the full production university network was restricted for ethical and operational reasons.

Additionally, the system is designed as a decision-support tool and does not implement automated vulnerability remediation or patch deployment. Final remediation actions remain dependent on human administrators.

Finally, resource constraints, such as limited computational resources and time, may limit the complexity of the AI models implemented. Advanced deep learning techniques may therefore be beyond the scope of the current implementation.

CHAPTER TWO

LITERATURE REVIEW

2.0 INTRODUCTION

This chapter covers existing literature on vulnerability management, cyber threats in a university network, traditional and AI-based vulnerability assessment approaches, and it discusses the research gaps addressed by the study.

2.1 Conceptual Review

2.1.1 Cybersecurity in University Networks

University networks are complicated systems that support a lot of work; they support academic, administrative, and research activities, consisting of users like staff and students. The nature of university network makes them easy targets for cyberattacks.

In several studies, educational institutions have been identified as attractive targets for cybercriminals due to the high volume of sensitive and heterogeneous data stored on their networks, including student records, financial information, and valuable research data (Abomhara & Køien, 2019; von Solms & van Niekerk, 2013; EDUCAUSE, 2022). , It is shown
that educational institutions are appealing targets for cyber criminals because of the data that can be gotten on the network.

In developing countries, Nigeria included, there are additional challenges for universities such as inadequate security funding, shortage of skilled cybersecurity personnel, and reliance on outdated systems, all of which significantly increase their vulnerability to cyber-attacks

(Adebayo & Abdulrahman, 2020; Adeyemo et al., 2021; World Bank, 2021)

2.1.2 Vulnerabilities and Vulnerability Management

Vulnerability can be defined as an inherent weakness in the computer system that allows hackers to attack the integrity, confidentiality, and availability of a system. Vulnerability management is

the ongoing processof identification, evaluation, prioritization, and remediation of vulnerabilities present within a system.

Vulnerability management is a process that is employed in cybersecurity for the purpose of identification of weaknesses that could affect a system in a harmful manner.

The lifecycle of vulnerability management is as follows.

1. Vulnerability discovery
2. Risk assessment
3. Prioritization
4. Remediation
5. Verification

If any failure occurs during the life cycle of vulnerability management, it could lead to unpatched systems and cyber risks increasing the chances of cyber attacks (Shah & Mehtre, 2020; Holm et al., 2015).

Vulnerability management is different from Intrusion detection; the major difference is that vulnerability management is proactive, as it focuses on preventing threats. While intrusion detection is reactive, it addresses threats while or after they have occurred.

Recent vulnerability management research calls for the adoption of the continuous intelligencedriven prioritization mechanism which combines technical severity with contextual and operational risk indicators, especially in the environments with limited cybersecurity resources such as academic institutions (Chen et al., 2022).

2.1.3 Vulnerability Scanning Tools

Traditional Vulnerability scanning tools are used to identify weaknesses on networks, applications or websites. These are a few of them.

1. OpenVAS (Open Vulnerability Assessment System): This is an open-source framework developed and maintained by the German cybersecurity company- Greenbone Networks. It's a free scanner that identifies weaknesses like outdated software, misconfigurations and weak passwords.
2. Nessus: This is an assessment tool by Tenable, it scans networks, systems, and applications and generates reports with vulnerability scorings to prioritize fixes.
3. Qualys: this is a cloud based platform that lets organizations perform vulnerability assessment.
4. OWASP ZAP (Zed Attack Proxy): is an open source tool for performing vulnerability scans in web applications.
5. Nmap (Network Mapper): A lightweight open-source tool used for network discovery and service version detection. It identifies live hosts, open ports, and running software versions, providing asset inventory that feeds into vulnerability matching.

These are all good for identifying weaknesses, but they often generate a large volume of alerts. Holm et al. (2015) wrote that the number of vulnerabilities discovered and the context provided often left IT security personnel overwhelmed when trying to process the results. This challenge is especially significant in resource-constrained areas like university networks.

2.1.4 Severity Scoring and Risk Assessment Models (Vulnerability Severity, CVSS) The Common Vulnerability Scoring System(CVSS) is a framework that is used to determine the severity of a vulnerability. CVSS assigns numerical scores based on base, temporal, and environmental metrics, which are intended to reflect the potential impact of a vulnerability.

Despite its widespread adoption, several studies have criticized CVSS for its inability to accurately represent real-world risk. Allodi and Massacci (2014, 2017) demonstrated that vulnerabilities with

the highest CVSS scores are not always exploited in practice, while some low- scoring vulnerabilities are often targeted. Similar findings were reported by Jacobs et al. (2021) and Holm et al. (2015), who observed a weak correlation between CVSS severity ratings and actual exploitation trends. Spring et al. (2021) further argued that CVSS focuses primarily on technical characteristics while neglecting contextual and operational factors, making CVSS-only prioritization insufficient for effective vulnerability management.

More recent studies provide further evidence supporting these findings as they prove that riskbased vulnerability scoring models which together with exploit likelihood and asset criticality outperform traditional severity-only approaches in real world environments (Almukaynizi et al., 2022; Ponta et al., 2020).

2.2 Theoretical Review

2.2.1 Risk Based Security Theory

This theory supports the need for intelligent prioritization mechanism that considers contextual factors. The theory states that actions made in cybersecurity should be decided based on risk and not just the availability of a vulnerability.

This approach supports the need for intelligent prioritization mechanisms that incorporate contextual and operational factors, enabling organizations to allocate limited security resources more effectively and focus on vulnerabilities that pose the greatest real-world risk (ENISA, 2021).

2.2.2 Attack Surface Theory

According to this theory, it is recommended that the number of vulnerable points in a system be capped Cite. If there are an appreciable number of vulnerable points in a system, it leads to a larger

attack surface being established by university networks. Understanding this is key to effective management of risk reduction.

2.2.3 Explainable Artificial Intelligence (XAI)

Explainable AI recommends transparency and interpretability in AI systems. Explainability is important because cybersecurity teams have to understand why a vulnerability is prioritized, in order for them to take the recommended action given by AI. Merging XAI into vulnerability management improves trust, accountability and usability. Techniques such as Shapley Additive Explanations (SHAP) and Local Interpretable Model-Agnostic Explanations (LIME) are widely used XAI methods that offer insight into how machine learning models arrive at specific predictions and rankings, thereby supporting informed decision-making in security operations (Lundberg & Lee, 2017; Ribeiro et al., 2016).

The most recent progress made in the field of explainable AI has emphasized the requirement for transparency and interpretability in operational cyber security systems. Analysts need to comprehend and have faith in the automated decisions so they can carry out the right remediation actions (Guidotti et al., 2019; Barredo Arrieta et al., 2020).

2.3 Empirical Review of Related Studies

Many research papers have gone into various ways of using AI and ML in cybersecurity.

Sommer and Paxson (2010) tried to determine the drawbacks of machine learning in intrusion detection systems and then context was highlighted as a significant factor in security decision making. Buczak and Guven (2016) reviewed all the ML methods that have been applied so far in intrusion detection, and they pointed out the very high success rate of these methods in detecting sophisticated attack patterns.

Allodi, Corradin, and Massacci (2021). proposed a data-driven approach that leverages realworld attack observations and vulnerability metadata to anticipate which vulnerabilities are likely to be exploited, demonstrating that external threat intelligence can enhance exploit prediction. The same with the Exploit Prediction Scoring System (EPSS), which was built by FIRST.org it calculates the likelihood of vulnerability exploitation in a certain period. All these methods will help to score vulnerabilities according to their dangers but still are not enough for overall vulnerability management.

According to Wang et al. (2022), the use of artificial intelligence for network threat detection through a honeypot data system was just one of the applications of their novel network threat detection system. The system managed to reach an impressive level of accuracy when it came to the detection of attacks, yet the major part of its work was for the detection of threats reactively rather than prioritization of vulnerabilities proactively.

2.4 Review of Existing Vulnerability Management Systems

The functionality and efficacy of vulnerability management and security systems that are currently in use are different (Holm et al., 2015; Wang et al., 2022). Classic vulnerabilities detectors like OpenVAS, Nessus, and Qualys are generally used, but they mainly depend on

CVSS-based prioritization. On the other hand, exploit prediction systems like the Exploit Prediction Scoring System (EPSS) predict the chance of exploitation, but they do not take into account the context of the organization or supply remediation workflows. Detection of intrusions is considered as an identifying process of malicious activities instead of a step to exploit

prevention.

At the same time, a new approach to AI-powered vulnerability management is being developed that promises to overcome these challenges. Such an approach would combine context-sensitive

risk evaluation, predicting the future use of exploits, and providing understandable support in decision-making, however, most of them still consider deployment on large enterprises as the main focus and do not take into account the specific operational limits of university networks (Chen et al., 2022; Ponta et al., 2020).Systems for intrusion detection based on AI have very high detection rates, but they cannot often explain their decisions or to assess risks proactively. These limitations emphasize the need of a solution for vulnerability management, that is integrated, aware of the context, and explainable, and that is designed specifically for the university environment.

2.5 Identified Research Gaps

The literature review identified the following gaps, presented in Table (2.1) below, alongside the studies that identified them and how this work addresses each gap.

Table 1: Identified Research Gaps

Research Gap	Studies Identifying the Gap	How This Work Addresses It
CVSS-only prioritisation does not reflect real-world exploitability	Allodi & Massacci (2014, 2017); Jacobs et al. (2021); Holm et al. (2015)	The system integrates EPSS exploit probability and CISA KEV data alongside CVSS, trained on 155,852 real CVEs
Lack of contextual awareness in existing tools	Chen et al. (2022); Ponta et al. (2020)	Asset criticality, network exposure, attack vector, and institutional context are incorporated as XGBoost model features
Reactive focus of AIbased security solutions	Sommer & Paxson (2010); Wang et al. (2022)	The proposed system is proactive — it prioritises vulnerabilities before exploitation occurs
No explainable, userfriendly systems for university environments	Guidotti et al. (2019); Barredo Arrieta et al. (2020)	SHAP explainability shows why each CVE was ranked; plain-English remediation is generated for non-expert IT staff
Insufficient research on vulnerability management in academic institutions	Adebayo & Abdulrahman (2020); Adeyemo et al. (2021)	This study focuses specifically on Nigerian university networks using a GOUNI simulated environment as a case study

2.6 Summary of Literature Review

Table 2 summary of literature review

Author(s) & Year	Focus Area	Key Finding	Limitation
Allodi & Massacci (2014, 2017)	Vulnerability exploitability vs. CVSS	High CVSS scores do not predict real-world exploitation	No complete management system proposed
Sommer & Paxson (2010)	ML in intrusion detection	Context is critical for security decisions	Reactive focus, not proactive
Buczak & Guven (2016)	ML in intrusion detection	ML methods achieve high detection rates	Not applied to vulnerability prioritization
Wang et al. (2022)	AI-based threat detection	High accuracy in detecting attacks via honeypot data	Reactive system, no proactive management
Allodi et al. (2021)	Exploit prediction	External threat intel improves exploit prediction	No remediation workflow
Chen et al. (2022)	Context-aware prioritisation	Contextual models outperform CVSS-only	Designed for enterprise, not universities
Ponta et al. (2020)	Risk-based assessment	Risk-based models outperform severity-only	No explainability for IT personnel
Jacobs et al. (2021)	Exploit Prediction Scoring System	EPSS predicts exploitation probability per CVE	Does not integrate asset context or remediation

The literature review has produced some gaps that need to be filled:

1. Vulnerability prioritization is based mostly on CVSS scoring that gives severity ratings.
2. Traditional vulnerability management systems lack contextual awareness.
3. AI based cybersecurity solutions still have a primarily reactive focus.

4. There are no explainable and user-friendly systems specifically designed for university environments.
5. Research on vulnerability management in academic institutions with limited resources is still not enough.

The above gaps bring out the fact that a university's network needs a specific AI-driven, contextaware, and explainable vulnerability management system.

In this chapter, the concepts, theories, and empirical studies relating to vulnerability management and AI in cybersecurity were reviewed. Table 2.2 below summarizes the key studies reviewed.

The review disclosed that the existing tools and methods did not have the capability to prioritize the vulnerabilities according to the actual environmental conditions and risk factors in that area. These deficiencies were the reason for the new AI-Powered Threat Assessment and Recommendation System that would serve as a tool for future-proof, understandable, and efficient vulnerability care in campus networks to be developed.

CHAPTER THREE

SYSTEM ANALYSIS AND DESIGN

3.0 Introduction

This chapter presents the system analysis and design of the AI-Powered Threat Assessment and Recommendation System. It begins with an analysis of the existing vulnerability management system used in university environments, identifying its operational workflow, limitations, and the gaps that justify the proposed solution. The chapter goes on to provide the requirements specifications for the system proposed, both functional and non-functional requirements. The methodology of design along with the method of data collection, is mentioned. Further, the models used in the system design have been explained with the help of UML models, namely the use case diagram, activity diagram and class diagram.

3.1 System Analysis

The system analysis phase involves analyzing a problem domain and determining the requirements, constraints, and areas that need improvement. In this chapter, we analyze both the current vulnerability management system that operates at Godfrey Okoye University and other Nigerian universities and the proposed artificial intelligence-based system.

3.1.1 Analysis of the Existing System

The current method used for managing vulnerabilities at most Nigerian universities, including Godfrey Okoye University, is a periodic and manual process, which makes use of conventional scanning systems and CVSS severity ratings. This process occurs as described below:

Figure 3.0-1 ***Workflow of the Existing Vulnerability Management Approach***

1. A vulnerability scanning program like Nessus or OpenVAS is scheduled or initiated by an IT administrator against certain parts of the university network.
2. The scanning program scans each host in the network to find out vulnerabilities in terms of version numbers of software programs, open ports and services configured on those hosts using a database of Common Vulnerabilities and Exposures (CVEs).
3. A report with all identified vulnerabilities is then prepared sorted according to their CVSS (Common Vulnerability Scoring System) base score which is a number between 0.0 and 10.0 depending upon technical severity criteria.
4. The IT administrator then manually reviews that report from the highest CVSS score to the lowest and tries to mitigate those vulnerabilities according to that score order.
5. External threat intelligence is not taken into account. The CVSS score alone is responsible for deciding the vulnerability to be patched up. The users of the current system include IT administrators and network managers who do not specialize in security issues but are generalists. The whole process is completely manual, periodic and does not generate any automatic remediation guidance.

3.1.2 Limitations of the Existing System

The comprehensive study of the current vulnerability management strategy showed the following significant shortcomings, which directly substantiate the need for developing the proposed solution:

I. Inaccuracy of CVSS-based priority setting. The current scanners give the highest priority to the vulnerability based on its CVSS value. Vulnerability rated CVSS 9.8 that was never used in the real attack can be given much more attention than vulnerability rated CVSS 6.5 that is actually exploited in real attacks. It has been proven by research conducted by Allodi and Massacci (2014), and Jacobs et al. (2019).

II. Absence of context awareness. In the current solution, no matter what the vulnerability is – whether it is affecting a mission-critical database server or an unimportant shared printer – it gets the same CVSS rating .

III. Alert fatigue from extensive lists of vulnerabilities. It is common practice for vulnerability scanners to generate several hundred or even several thousand results in one scanning cycle. If not filtered intelligently, IT employees become overloaded and cannot pay attention to some critical vulnerabilities among a great number of alerts (Holm et al., 2011).

IV. Lack of guidance for remediation in non-technical language. Current tools provide technical information about CVEs which should be understood by a person who has the knowledge in cybersecurity field. IT employees at universities, who are not cybersecurity specialists, get no guidance about actions to take next.

V. Lack of transparency of decision-making process. IT administrators are forced to accept the CVSS-based priority ranking without having an explanation of what was the reason behind the score.

VI. Intended for use in enterprise environment. All the advanced vulnerability management systems are designed for big enterprises and are priced accordingly. Such products are not available for Nigerian universities due to lack of budget for IT security and small size of IT department.

VII. Reactive and intermittent posture. The current approach is manual and occasional. The process does not include any live threat intelligence such as EPSS scores or CISA Known Exploited Vulnerabilities (KEVs), which means that the system cannot adjust to any exploitation attempts made in the wild.

3.1.3 Analysis of the Proposed System

The Proposed AI-Powered Threat Assessment and Recommendation System solves all limitations of the current system in the form of a smart, context-aware, and explainable vulnerability management platform. The proposed system provides the following capabilities:

I. Combines the only CVSS-based ranking with the two-layer AI ranking system. In the first layer, the binary XGBoost model is used for predicting which vulnerabilities will be actively exploited based on such inputs as EPSS, CVSS, exploit techniques, and age of vulnerabilities. The second layer combines the output of the AI system with the context information about assets and their network to create the final ranked list.

II. Enables integrating real-world threat intelligence provided by three reliable sources – exploit probability score EPSS from FIRST.org, CISA Known Exploited Vulnerabilities catalog (proof of active exploits), and NVD CVE details.

III. Generates easy to understand remediation recommendations in plain language using a keyword-based NLP engine, where each vulnerability classified into one of 11 categories and corresponding recommendation created with urgency labels (Immediate, Soon, and Routine).

IV. Provides explainable AI decisions using the SHAP technique (Shapley Additive Explanations) to allow IT administrators understand which features were responsible for each priority ranking of the vulnerabilities.Runs entirely on standard university hardware with no cloud infrastructure, no licensing fees, and a simple browser-based interface accessible to non-specialist IT staff.

V. Takes Nmap XML scan results as an input where IT administrators can upload network scans through the web portal and get the AI ranked results within seconds.

3.2 System Requirements Specification (SRS)

The Functional and non-functional requirements are listed in the System Requirements Specification document, which explains what should be done by the new system. Such requirements were obtained based on an analysis of limitations in the current system and research objectives described in Chapter 1.

3.2.1 Functional Requirements

The Functional requirements state the particular features that the system should be able to offer:

I. Accept Nmap XML files through a web browser and parse the data into structured asset entries which will contain IP addresses, hostnames, open ports, services, and versions of software found.

II. The system shall load and store CVE records from the Kaggle CVE/EPSS/KEV enriched dataset and the NVD API, including CVSS scores, EPSS exploit probability scores, and CISA KEV flags.

III. The system shall match discovered software versions on network assets to CVE records and generate a list of asset-vulnerability pairs representing the network's exposure.

IV. The system shall compute an AI-based exploitability probability score for each CVE using an XGBoost binary classifier trained on 155,852 real CVEs.

V. The system shall compute a final composite priority score for each asset-CVE pair by combining the AI exploitability probability (60%), CVSS normalised score (25%), and asset criticality (15%).

VI. The system shall rank all asset-vulnerability pairs by composite priority score and assign tier labels: CRITICAL, HIGH, MEDIUM, or LOW.

VII. The system shall classify each CVE into one of 11 vulnerability categories using keyword-based NLP and generate plain-English remediation recommendations with urgency labels.

VIII. The system shall provide SHAP-based explanations for each vulnerability ranking, showing the contribution of each input feature to the risk score.

IX. The system shall display all results through an interactive web dashboard with five pages: main dashboard, assets overview, asset detail, vulnerability detail, and evaluation page.

X. The system shall allow filtering and keyword search of the ranked vulnerability list.

XI. The system shall provide a downloadable CSV export of the full risk-ranked vulnerability report.

XII. The system shall provide a performance evaluation module comparing AI ranking against CVSS-only baseline using Precision@K, Recall@K, MAP, and NDCG@K metrics.

3.2.2 Non-Functional Requirements

Non-functional requirements determine the required qualities and attributes that the system needs to possess:

Usability: Dashboard should be usable by IT professionals without any specific skills in cybersecurity. All pages should be compliant with Shneiderman's Eight Golden Rules of Interface Design. Each page should have a back navigation button to allow easy reversibility. All texts should be readable with sufficient contrast ratios per WCAG rules.

Performance: Main dashboard should load within 3 seconds for at least 400 vulnerabilities on standard hardware. ML scoring pipeline should process 155,852 CVEs and assign risk scores in 2 minutes. API should respond to most requests within 500 milliseconds.

Security: All vulnerabilities' data should be kept locally in SQLite database. No network sensitive data should be transferred to external servers. All file uploads should be validated before any operations with the file.

Reliability: The system should react to missing EPSS scores or KEV data with neutral default values, avoiding failure of the pipeline due to external data absence.

Portability: The program should run under the Windows, macOS, and Linux operating systems using solely native Python modules and a contemporary web browser. Cloud services, database, or any proprietary software shall not be used.

Scalability: The layered structure shall ensure that each module can be independently modified or replaced without interfering with other layers

3.3 System Design Methodology

The entire system has been designed and developed using the combination of the Object-

Oriented Analysis and Design (OOAD) methodology and an Agile approach to development. OOAD has been used in the analysis and design phase to represent the components of the system, their relations and interactions through UML diagrams, while the agile process controlled the implementation phase.

3.3.1 Justification for Methodology

OOAD was chosen based on the following reasons:

I. Modularity: the OOAD approach leads to creation of a modular system, where every single component of it (the data collection, ML scoring, NLP engine, REST API, and the dashboard) can be created, tested and updated independently. It is vital for integration of multiple AI components in the system.

II. Clearness of the system components: using UML diagrams allows for visualization of the system's structure and behavior in a way understandable for everybody – the developer, the supervisor and future maintainers of the system.

III. Reusability: the object-oriented design allows classes and modules to be reused between different components. For instance, the feature engineering module is used both in the training pipeline and the real-time scoring pipeline.

IV. Compatibility with the technology stack: the main programming language of implementation is Python, which is an entirely object-oriented language. Similarly, Flask, XGBoost, and SQLite have object-oriented APIs.

Agile methodology has been chosen for this project because there were changes in the requirements of the system throughout the whole project cycle based on the results of the evaluation process. The iterative nature of the development allowed to adjust the model and

architecture (particularly, to change the type of the model from regression to classification and split the training and scoring data sets).

3.3.2 Method of Data Collection

The collection of data for the study was carried out in the following way:

Data on network assets was obtained by performing active network scanning with the Nmap (Network Mapper) tool on the VMware simulated environment of the university network. There were eight virtual machines created in the VMware lab specifically configured to emulate typical types of network assets of universities: web server with Apache, database server with MySQL, two workstations, mail server, student portal server, FTP server, and network gateway. Nmap was run with options for service version (-sV) and OS detection (-O), which resulted in creating an XML file that was uploaded to the system. Scanning was done in the simulated VMware environment. No live production network systems were scanned.

Data on vulnerabilities and threats was obtained from three secondary sources: the Kaggle CVE/EPSS/KEV Enriched Dataset (with 155,852 real CVE records obtained in the form of CSV file); the NIST National Vulnerability Database (NVD) REST API v2 (CVE descriptions and CVSS scores were obtained programmatically); and the FIRST.org EPSS API (for exploit probability scores). These sources did not require any special access permissions.

3.4 System Modelling Using UML

The UML diagrams were used to formally represent the system's functionalities, behaviors, and structures. There were three main types of UML diagrams created: a use case diagram representing the interaction between actors and the system, an activity diagram representing the workflow process of vulnerability assessment, and a class diagram representing the system's object structure.

3.4.1 Use Case Diagram

The use case diagram identifies the actors who interact with the system and the use cases (system functions) they trigger. Two actors were identified:

I. Primary Actor, IT Administrator: The university IT staff member responsible for network security, who uploads Nmap scan files, views ranked vulnerability reports, reads remediation advice, and exports results.

II. Secondary Actors, External Data Sources: Nmap scanner (provides asset discovery data); Kaggle dataset/NVD API (provides CVE data); EPSS API (provides exploit probability scores); CISA KEV catalog (provides confirmed exploitation flags).

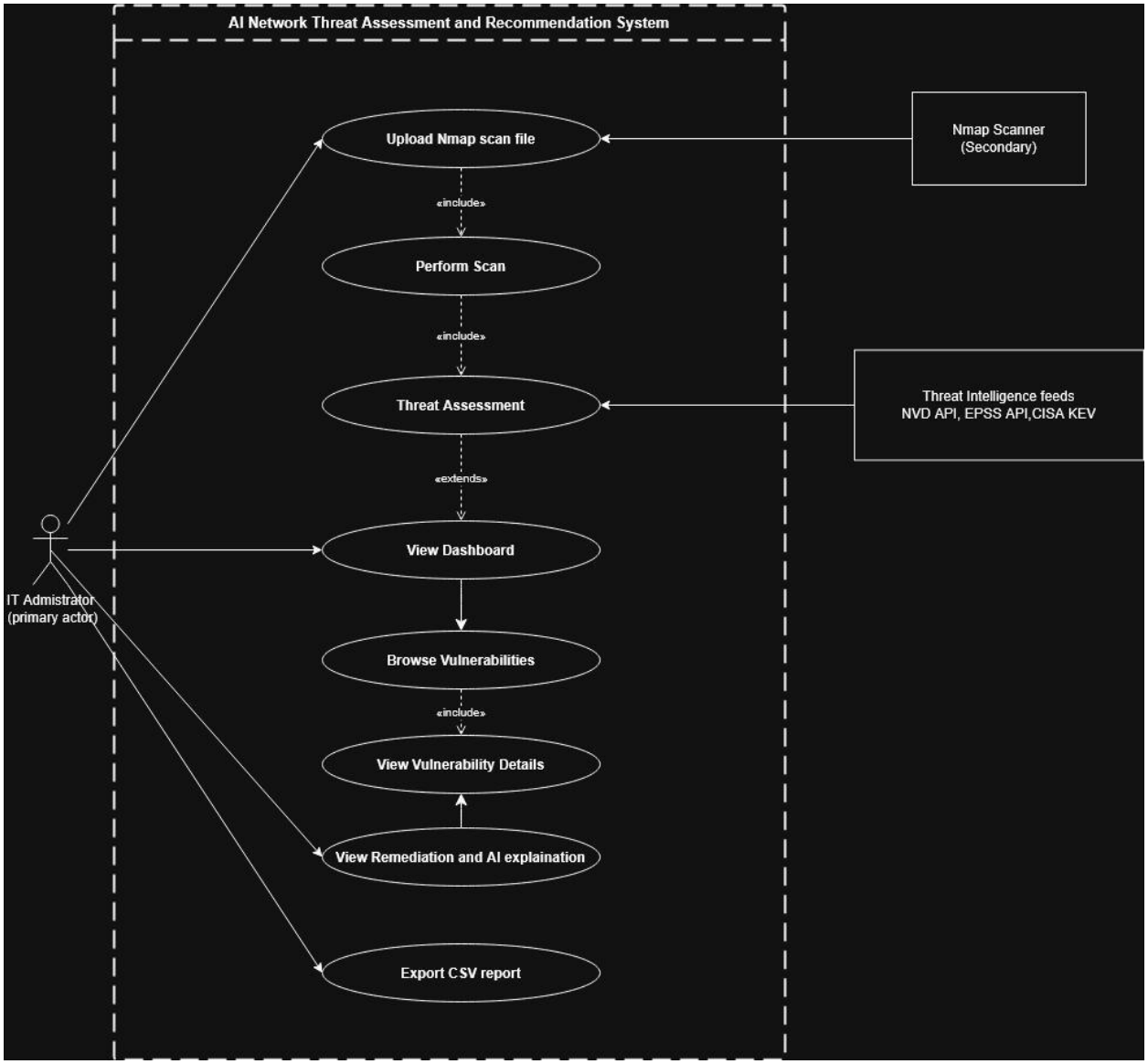


Figure 3.0-3 Use Case Diagram of th AI Network Threat Assessment and Recommendation System

Table 3.1 presents the main use case specification for the system.

Table 3.1: Main Use Case Specification — AI Vulnerability Assessment

Use Case Name:	AI Vulnerability Assessment and Prioritization
Use Case ID:	UC-01
Actor:	IT Administrator (Primary)
Trigger:	IT Administrator uploads a Nmap XML scan file through the web dashboard.
Description:	The system accepts a Nmap XML scan file, parses all discovered assets and services, matches them against the CVE database, computes AI-based risk scores using the XGBoost classifier, generates plain-English remediation advice, and displays a ranked vulnerability list on the dashboard.
Preconditions:	(1) The system database is populated with CVE, EPSS, and CISA KEV data. (2) The IT Administrator has completed a Nmap scan and has the XML output file. (3) The ML model has been trained and saved.
Normal Course:	(1) IT Administrator navigates to the Upload Scan page. (2) The file is validated as a .xml Nmap file. (3) Extraction of assets and services from XML. (4) Matching of asset software with their corresponding CVEs. (5) XGBoost scoring of all asset-CVE combinations. (6) Remediation suggestions produced by NLP engine. (7) Dashboard displays ranked results.
Post-conditions:	The database is updated with new asset records and risk scores. The dashboard displays the current ranked vulnerability list. The IT Administrator can filter, search, and export results.
Exceptions:	E1: Invalid file type — System displays error: "Please upload a valid Nmap XML file." E2: No live hosts found — System displays: "No active hosts found. Ensure hosts are running during the scan."

3.4.2 Activity Diagram

The activity diagram depicts the order of activities involved in the process of vulnerability assessment starting from the upload of Nmap scan through the presentation of ranked results on the dashboard.

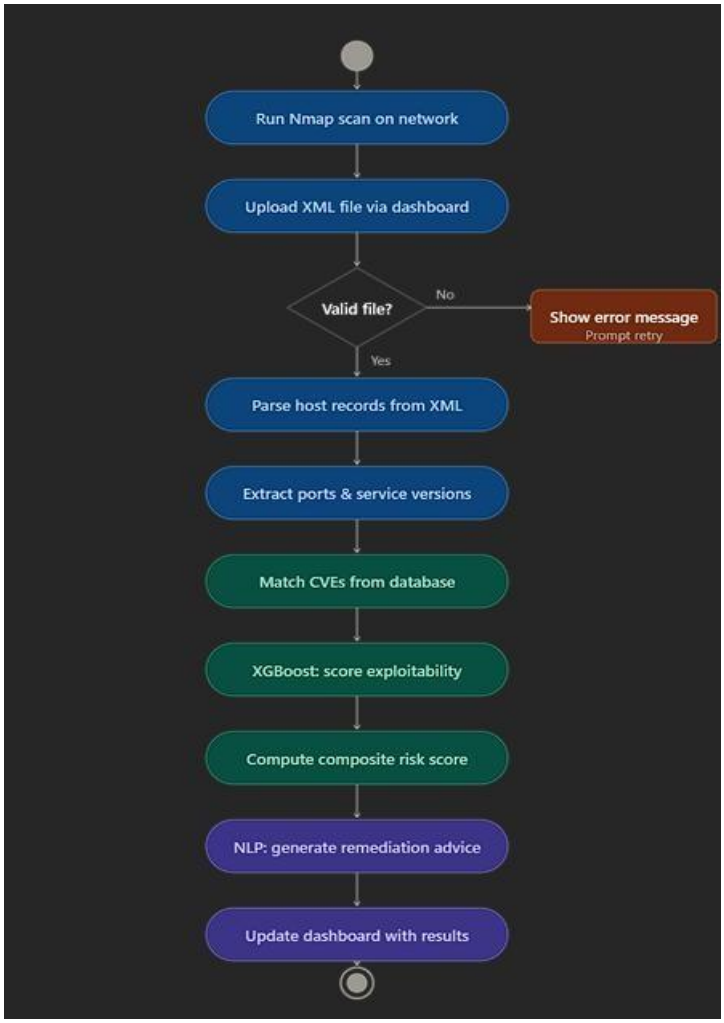


Figure 3.4 ***Activity Diagram of the Vulnerability Assessment Process*** The process flow of activity is as following: IT Administrator performs a scan of the network with Nmap and uploads the produced XML file. The system checks if the file is valid and parses all host entries if it is. Open ports and version of the services for each host are identified. Vulnerability database is queried for the CVEs that match discovered software. XGBoost classifier calculates exploitability

probability for each CVE found. Risk fusion component evaluates composite priority score based on the AI score and asset criticality. NLP component produces remediation recommendations for each class of vulnerabilities. Data is saved to the database and dashboard is updated with the ranked list. In case of any problems with the file being uploaded or no hosts found, system reports appropriate error and asks IT

Administrator to try again.

3.4.3 Class Diagram

The class diagram displays the classes in the system, their properties and their relationships. It was observed that there are seven main classes, which correspond to the six database tables and their respective processing modules.

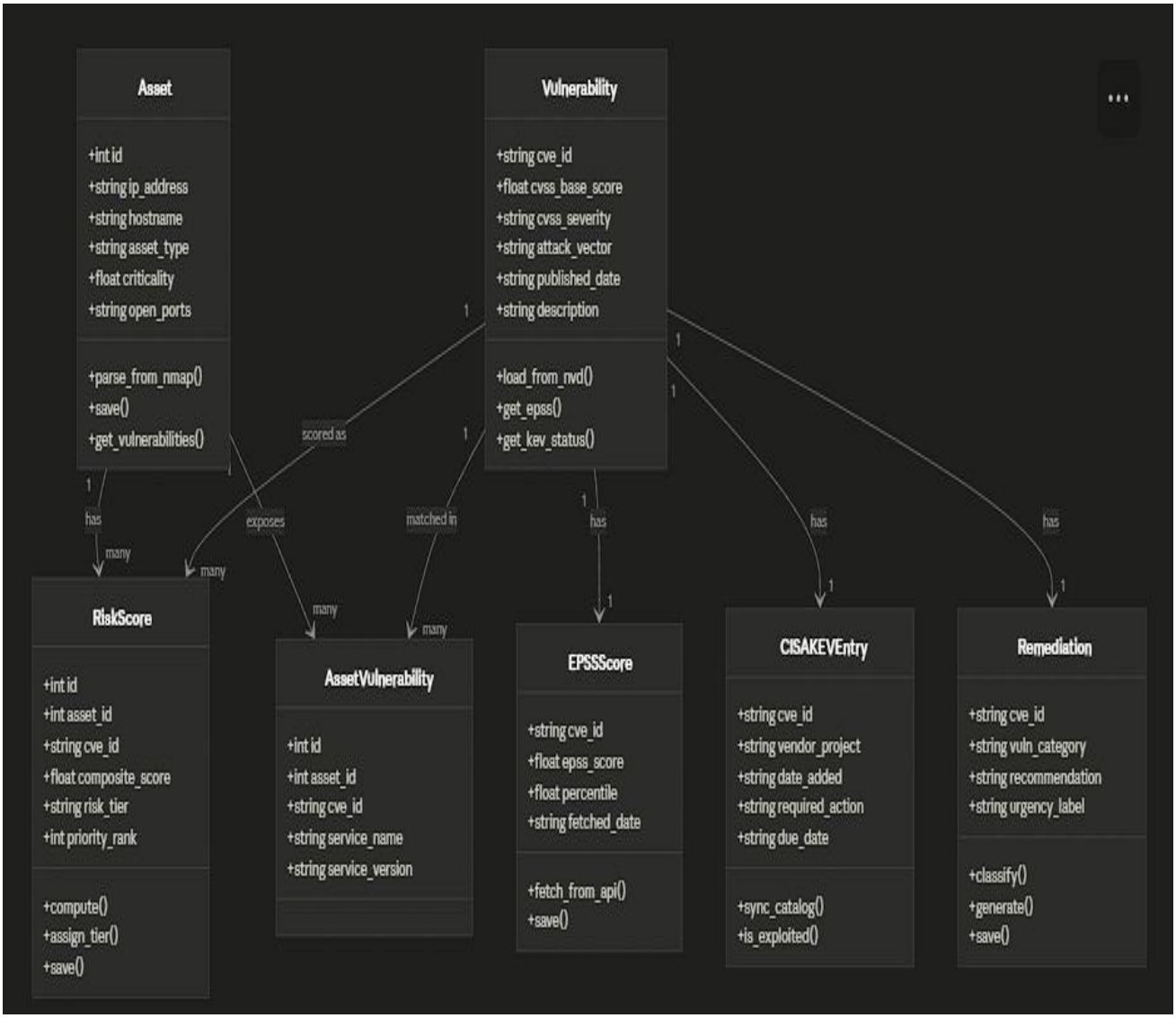


Figure 3.5: Class Diagram of the AI Threat Assessment and Recommendation System

Class	Key Attributes	Key Methods	Relationships
Asset	id, ip_address, hostname, asset_type, criticality, open_ports	parse_from_nmap(), save(), get_vulnerabilities()	One Asset has any AssetVulnerability; ıe Asset has any RiskScore
Vulnerability	cve_id, cvss_base_score, cvss_severity, attack_vector, published_date, description	load_from_nvd(), get_epss(), get_kev_status()	One Vulnerability has one EPSSScore; one Vulnerability has one CISAKEVEntry; one Vulnerability has one Remediation
EPSSScore	cve_id, epss_score, percentile, fetched_date	fetch_from_api(), save()	Belongs to one Vulnerability
AssetVulnerability	id, asset_id, cve_id, service_name, service_version	match(), save()	Belongs to one Asset and one Vulnerability
CISAKEVEntry	cve_id, vendor_project, date_added, required_action, due_date	sync_catalog(), is_exploited()	Belongs to one Vulnerability
RiskScore	id, asset_id, cve_id, composite_score, risk_tier, priority_rank	compute(), assign_tier(), save()	Belongs to one Asset and one Vulnerability
Remediation	cve_id, vuln_category, recommendation, urgency_label	classify(), generate(), save()	Belongs to one Vulnerability

3.5 System Design

The system design process is where the results of analysis and modeling become actual plans for implementation. In this chapter, we discuss input design, output design, database design, and system design.

3.5.1 Input Design

Inputs to the system are of two types: those from the IT Administrator via the web interface, and those from external APIs/files. Inputs from the IT Administrator using the web interface are:

I. Nmap XML scan file: This is uploaded using the drag and drop file upload page. It is verified as an .xml file before processing. Templates for the Nmap commands, with copy functionality, are available to the administrator to help them create the required format.

II. Filters used: Tier filter buttons (All/Critical/High/Medium/Low) and keyword search on the dashboard for filtering vulnerabilities.

III. Re-score button: The button that calls the ML pipeline to score all vulnerabilities using the current data.

    a. External data inputs include:

IV. Kaggle CSV Dataset: This dataset is uploaded only once during setup and contains 155,852 CVE records with CVSS, EPSS, and KEV data..

V. NVD REST API: queried for supplementary CVE descriptions and CVSS metrics.

VI. EPSS API (FIRST.org): queried for current exploit probability scores.

VII. CISA KEV JSON feed: downloaded periodically to update the confirmed exploitation list.

    a. All the input from the user is checked prior to any processing. The file type is checked before the uploading process. Malformed XML files cause an informative

error message to be displayed, indicating that the user needs to conduct another scan using the right flags.

3.5.2 Output Design

The following is generated by the system:

I. AI-Prioritised Vulnerability List: rendered on the dashboard as a paginated and filterable table listing the priority rank, CVE ID, risk tier (coloured badge), AI score (bar chart), CVSS score, EPSS probability, KEV indicator, vulnerable host IP address, category of vulnerability and urgency level. The rows can be clicked to link to the full details of the vulnerability.

II. Vulnerability Details Page: renders the complete CVE description, bar chart representation of the four aspects of AI scoring, viz., CVSS contribution, EPSS contribution, KEV contribution, asset criticality contribution; the CVSS metric dimensions, viz., attack vector, complexity, privileges, user interaction, impact on confidentiality, integrity, availability, remediation advice in plain English with urgency level and the list of all affected network assets.

III. Asset Details Page: displays all the vulnerabilities affecting a particular host sorted by AI scores, total count of tiers per host and the list of ports opened and services running.

IV. Evaluation Page: displays the live Precision@K and Recall@K comparison chart between the AI prioritisation system and CVSS-only prioritisation system along with the complete metrics table including AUC, MAP, NDCG, F1-score, confusion matrix and rank improvement statistics.Evaluation Page: displays live Precision@K and Recall@K charts comparing the AI system against CVSS-only prioritisation, the full metrics table (AUC, MAP, NDCG, F1, confusion matrix), and rank improvement statistics.

V. CSV Export: a downloadable spreadsheet comprising all ranked vulnerabilities together with their scores, tiers, CVSS, EPSS, KEV tags, hosts, remediations, and urgencies. Ideal for presenting to university management or auditors.

3.5.3 Database Design

The database used is a SQLite relational database stored in a single local file (vulnerabilities.db). SQLite is preferred over a client-server database like MySQL or PostgreSQL since it does not require a database server to be installed, runs on all platforms, and is enough for the amount of data needed for a vulnerability management system at a university.

Table 3.2 presents the database schema.

Table: assets				
Column Name	**Data Type**	**Size**	**Constraints**	**Description**
id	INTEGER	4 bytes	PRIMARY KEY, AUTOINCREMENT	Unique identifier for each discovered asset
ip_address	TEXT	max 45 chars	NOT NULL, UNIQUE	IPv4 or IPv6 address of the network host
hostname	TEXT	max 255 chars	NULLABLE	Resolved hostname or device name
asset_type	TEXT	max 50 chars	NOT NULL	Category of asset (server, workstation, router, etc.)
criticality	REAL	8 bytes	NOT NULL, DEFAULT 1.0	Asset criticality weight used in risk scoring (0.0–1.0)
open_ports	TEXT	max 1000 chars	NULLABLE	JSON array of open ports and protocols from Nmap
last_scanned	TEXT	20 chars	NULLABLE	ISO 8601 timestamp of the most recent vulnerability scan

Table: vulnerabilities				
Column Name	**Data Type**	**Size**	**Constraints**	**Description**
cve_id	TEXT	max 20 chars	PRIMARY KEY	CVE identifier (e.g., CVE-2023-1234)
cvss_base_score	REAL	8 bytes	NULLABLE	CVSS v3.x base score in range 0.0–10.0
cvss_severity	TEXT	max 10 chars	NULLABLE	CVSS severity label (None / Low / Medium / High / Critical)
attack_vector	TEXT	max 20 chars	NULLABLE	CVSS attack vector (Network, Adjacent, Local, Physical)
attack_complexity	TEXT	max 10 chars	NULLABLE	CVSS attack complexity (Low or High)
privileges_required	TEXT	max 10 chars	NULLABLE	Privileges needed to exploit (None, Low, High)
published_date	TEXT	10 chars	NULLABLE	Date the CVE was publicly disclosed (YYYY-MM-DD)
description	TEXT	max 5000 chars	NULLABLE	Full vulnerability description sourced from NVD

Table: epss_scores				
Column Name	**Data Type**	**Size**	**Constraints**	**Description**
cve_id	TEXT	max 20 chars	PRIMARY KEY, FK → vulnerabilities	CVE identifier (foreign key)
epss_score	REAL	8 bytes	NOT NULL	EPSS probability of exploitation within 30 days (0.0–1.0)
percentile	REAL	8 bytes	NULLABLE	EPSS percentile rank among all scored CVEs (0.0–1.0)
fetched_date	TEXT	10 chars	NOT NULL	Date the EPSS score was retrieved (YYYY-MM-DD)

Table: cisa_kev				
Column Name	**Data Type**	**Size**	**Constraints**	**Description**
cve_id	TEXT	max 20 chars	PRIMARY KEY, FK → vulnerabilities	CVE identifier (foreign key)
vendor_project	TEXT	max 100 chars	NULLABLE	Affected vendor or product name from CISA catalogue
date_added	TEXT	10 chars	NOT NULL	Date CISA added the CVE to the KEV catalogue (YYYYMM-DD)
required_action	TEXT	max 500 chars	NULLABLE	CISA-mandated remediation action description
due_date	TEXT	10 chars	NULLABLE	CISA remediation compliance deadline (YYYY-MM-DD)

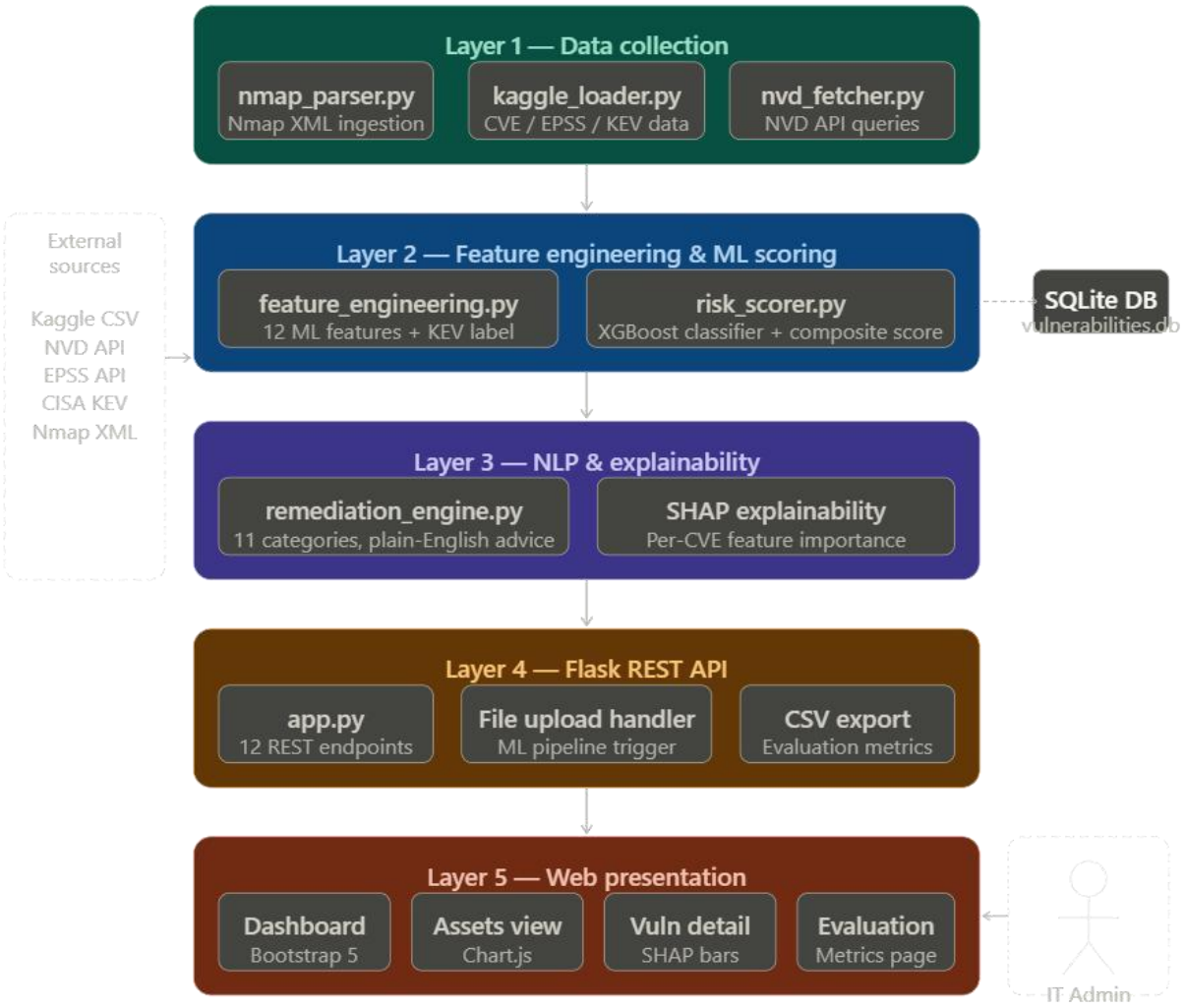
Table: asset_vulnerabilities				
Column Name	**Data Type**	**Size**	**Constraints**	**Description**
id	INTEGER	4 bytes	PRIMARY KEY, AUTOINCREMENT	Unique record identifier
asset_id	INTEGER	4 bytes	NOT NULL, FK → assets	Reference to the affected asset
cve_id	TEXT	max 20 chars	NOT NULL, FK → vulnerabilities	Reference to the matched CVE
service_name	TEXT	max 100 chars	NULLABLE	Name of the vulnerable service (e.g., Apache, OpenSSH)
service_version	TEXT	max 50 chars	NULLABLE	Detected version string of the vulnerable service

Table: risk_scores				
Column Name	**Data Type**	**Size**	**Constraints**	**Description**
id	INTEGER	4 bytes	PRIMARY KEY, AUTOINCREMENT	Unique record identifier
asset_id	INTEGER	4 bytes	NOT NULL, FK → assets	Reference to the scored asset
cve_id	TEXT	max 20 chars	NOT NULL, FK → vulnerabilities	Reference to the associated CVE
composite_score	REAL	8 bytes	NOT NULL	AI-computed composite risk score (0.0–100.0)
risk_tier	TEXT	max 15 chars	NOT NULL	Risk classification (Critical / High / Medium / Low)
priority_rank	INTEGER	4 bytes	NOT NULL	Ordinal priority rank across all asset-CVE pairs
computed_date	TEXT	20 chars	NOT NULL	ISO 8601 timestamp when the score was calculated

Table: remediations				
Column Name	**Data Type**	**Size**	**Constraints**	**Description**
cve_id	TEXT	max 20 chars	PRIMARY KEY, FK → vulnerabilities	CVE identifier (foreign key)
vuln_category	TEXT	max 50 chars	NULLABLE	Vulnerability category (e.g., RCE, SQLi, Misconfiguration)
recommendation	TEXT	max 2000 chars	NOT NULL	NLP-generated remediation advice in plain language
urgency_label	TEXT	max 20 chars	NOT NULL	Urgency classification (Immediate / Short-term / Routine)
patch_url	TEXT	max 500 chars	NULLABLE	URL to the vendor patch or security advisory

3.5.4 System Architecture Design

The system follows a five-layer web-based architecture deployed as a local web server. Figure 3.7 presents the system architecture diagram.



[Figure 3.7: System Architecture Diagram — AI Threat Assessment and Recommendation System]

The five layers are:

i. Data Collection Layer: Handles network asset discovery via Nmap XML parsing, CVE data ingestion from the Kaggle dataset and NVD API, EPSS score retrieval, and CISA KEV synchronization. This layer populates the SQLite database through the kaggle_loader.py, nmap_parser.py, and nvd_fetcher.py modules.

ii. Feature Engineering and ML Layer: The feature_engineering.py module extracts 12 features from raw database records and constructs the binary training label. The risk_scorer.py module trains the XGBoost binary classifier on 155,852 CVEs using a stratified 70/30 split, evaluates it on the held-out test set, and writes composite risk scores back to the risk_scores table.

iii. NLP and Explainability Layer: The remediation_engine.py module classifies each CVE into one of 11 vulnerability categories using keyword pattern matching and generates plain-English remediation advice. SHAP values are computed for the top-ranked vulnerabilities to provide per-CVE feature importance explanations.

iv. Flask REST API Layer: The app.py module exposes 12 REST API endpoints that serve all dashboard data as JSON. The API handles file uploads, triggers the ML pipeline, provides evaluation metrics, and streams the CSV export.

v. Web Presentation Layer: Five HTML pages built with Bootstrap 5 and Chart.js consume the REST API using JavaScript and render the interactive dashboard. All pages have a uniform navigation menu and stylesheet (main.css). ensuring interface consistency.

Table 3.3 summarizes the system architecture.

3.3: System Architecture Summary

Layer	Component	Technology	Function
Data Collection	kaggle_loader.py, nmap_parser.py, nvd_fetcher.py	Python, Requests	Ingests CVE data and network scan results into SQLite
Feature Engineering	feature_engineering.py	Python, Pandas, NumPy	Constructs 12 ML features and binary KEV label
ML Scoring	risk_scorer.py	XGBoost, Scikit-learn	Trainsclassifier; computes risk scores and tiers
Explainability	SHAP layer in risk_scorer.py	SHAP 0.44	Generates per-CVE feature importance explanations
NLP Remediation	remediation_engine.py	Python (custom NLP)	Classifies CVEs; generates remediation advice
REST API	api/app.py	Flask 3.0	Serves all dashboard data via 12 JSON endpoints
Web Dashboard	dashboard/templates/*.html	HTML5, Bootstrap, Chart.js	Interactive five-page vulnerability management UI
Database	vulnerabilities.db	SQLite 3	Local relational database; no server required

CHAPTER FOUR

SYSTEM IMPLEMENTATION

4.0 Introduction

In this chapter, the implementation of the AI-Powered Threat Assessment and Recommendation System will be discussed. This chapter will talk about the development environment and tools that have been used, dataset information and data pre-processing, machine learning model development and optimization, implementation of the system modules, and testing and evaluation of the system. In the end, the results will be presented and discussed, including interface screenshots and performance comparison with CVSS based vulnerability prioritization system.

4.1 Development Environment and Tools

The software was developed completely using an ordinary Windows 10 workstation without having any requirements for specific hardware or cloud-based infrastructure.

4.1.1 Programming Language and Libraries

For developing the program, Python 3.11 was chosen as the main programming language because of its large number of libraries related to data science, machine learning, and web technologies, wide usage in the field of cybersecurity, and easy readability by IT professionals.

Table 4.1 presents the complete list of libraries and frameworks used in the implementation.

4.1: Libraries and Frameworks Used in System Implementation

Library / Framework	**Version**	**Purpose**
Flask	3.0.0	Backend web framework — serves the REST API and web dashboard
XGBoost	2.0.3	Primary ML classifier for vulnerability exploitability prediction
Scikit-learn	1.3.2	RandomForest comparison model, train/test split, cross-validation, evaluation metrics
SHAP	0.44.0	Explainability layer — Shapley value computation for per-CVE feature importance
Pandas	2.1.4	Dataset loading, preprocessing, and feature engineering
NumPy	1.26.2	Numerical operations and array processing
SQLite (built-in)	3.x	Local relational database — stores assets, CVEs, scores, and remediations
Requests	2.31.0	HTTP client for NVD API, EPSS API, and CISA KEV catalog
Matplotlib	3.8.2	Evaluation chart generation — precision/recall curves, confusion matrix
Bootstrap	5.3.0	Frontend CSS framework for the web dashboard
Chart.js	4.4.0	Interactive charts on the dashboard (doughnut, bar, line charts)
Bootstrap Icons	1.11.0	Icon library for the user interface

4.1.2 Development Platform and Hardware Requirements

The system was built and trained on a Windows 10 desktop computer equipped with the following hardware specifications: Intel Core i5 CPU, 8GB RAM, 256GB SSD drive. No graphics processing unit was required because the XGBoost model can train itself quickly on a CPU for

the used dataset. The programming environment was Visual Studio Code with Python extension. This system will work on any common university workstation with the same minimum hardware specification.

Virtual environments were used in Python (venv), in order to manage project dependencies and make it reproducible on different computers. The full list of dependencies is described in the requirements.txt file of the project that accompanies the source code.

4.2 Dataset Description and Preprocessing

The dataset used for the training and testing of the artificial intelligence system was the publicly available CVE/EPSS/KEV Enriched Dataset downloaded from Kaggle website. It combines three datasets in one structured file, which makes it appropriate for vulnerability prioritization research based on machine learning techniques.

Table 4.2 describes the dataset characteristics.

4.2: Dataset Description — CVE/EPSS/KEV Enriched Dataset

Attribute	**Value**
Source	Kaggle — publicly available academic dataset
Total records	155,852 CVE entries
Features per record	17 columns (cve_id, base_severity, base_score, exploitability_score, impact_score, epss_score, epss_perc, cisa_kev, attack_vector, attack_complexity, privileges_required, user_interaction, scope, confidentiality_impact, integrity_impact, availability_impact, published_date)
CISA KEV positive class	963 CVEs (0.63% of total)
Non-KEV negative class	154,889 CVEs (99.37% of total)
EPSS score range	0.001 to 0.9735
EPSS mean score	0.0277
CVSS severity distribution	CRITICAL: 24,153 (15.5%) | HIGH: 61,680 (39.6%) | MEDIUM: 67,272 (43.2%) | LOW: 2,747 (1.7%)
Missing values	None — dataset is fully populated

4.2.1 Data Preprocessing Steps

The following data preprocessing steps were done on the dataset before the feature engineering step:

1. Null value check: There were no null values found among all 155,852 rows; therefore, no data imputation was necessary.
2. Type normalization: The string types columns (attack vector, attack complexity, privileges required, user interaction, and impact columns) were normalized to upper case and stripped white spaces.
3. Clipping of scores: The CVSS base score was clipped within the range of 0.0 to 10.0. The EPSS scores were clipped within the range of 0.0 to 1.0 to prevent out-of-range values because of some outlier entries.

4. Binary encoding: The cisa_kev field is converted from the string type "True"/"False" to integer binary type 1/0. This will become the ground truth label of the classification problem.
5. CVSS normalization: The CVSS base score was normalized to the 0.0 to 1.0 range by dividing the base score by 10.0.
6. Calculating vulnerability age: The published_date field was parsed and normalized to the vuln_age_days_norm which is number of days between the date of the publication and now divided by 1,095 (three years); clipped at 1.0. It reflects the intuition that older unpatched CVEs had enough time to develop exploits.
7. Construction of feature matrix: For the machine learning model, 12 features were identified. The in_cisa_kev was deliberately left out of the feature matrix, acting only as the target label so as to avoid target leakage.

4.3 Model Architecture and Training

The AI risk scoring model uses the XGBoost Binary Classifier model. XGBoost was chosen for its ability to always outperform other decision tree models in tabular data, class imbalance handling through scale_pos_weight and built-in SHAP explainability.

4.3.1 Feature Engineering

Twelve features were engineered from the raw dataset. Table 4.3 presents the full feature set.

4.3: Engineered Feature Set for the XGBoost Classifier

Feature Name	Type	Source	Description
cvss_normalized	Continuous	Kaggle/NVD	CVSS base score divided by 10.0
epss_score	Continuous	FIRST.org EPSS	Probability of exploitation within 30 days (0–1)
epss_percentile	Continuous	FIRST.org EPSS	Relative rank of CVE among all CVEs by EPSS
asset_criticality	Continuous	Nmap + config	Importance of affected host (0–1); neutral 0.5 at global training
attack_vector_net	Binary	Kaggle/NVD	1 if attack vector is NETWORK; 0 if LOCAL or PHYSICAL
attack_complexity_low	Binary	Kaggle/NVD	1 if attack complexity is LOW (easy to exploit)
privs_required_none	Binary	Kaggle/NVD	1 if no authentication required to exploit
user_interaction_none	Binary	Kaggle/NVD	1 if no victim interaction needed
vuln_age_days_norm	Continuous	NVD published_date	Age since publication, normalized over 3 years
c_impact_high	Binary	Kaggle/NVD	1 if confidentiality impact is HIGH
i_impact_high	Binary	Kaggle/NVD	1 if integrity impact is HIGH
a_impact_high	Binary	Kaggle/NVD	1 if availability impact is HIGH

Note: Inclusion of the in_cisa_kev feature (KEV membership) in the features table was intentionally avoided. This is because it provides the binary label for the model. Use of the feature in the set of inputs for prediction means that there will be a problem with target leakage where the model will perfectly predict its own label.

4.3.2 Training Configuration and Hyperparameters

All 155,852 CVEs in the Kaggle dataset were used to form the training dataset. A stratified train/test split in the ratio of 70/30 with positive class ratio of 0.63% was applied to both splits. The train dataset contained 108,596 CVEs with 675 KEV positives, while the test dataset contained 47,256 CVEs with 288 KEV positives..

Table 4.4 presents the XGBoost hyperparameters used in the final trained model.

Table 4.4: XGBoost Classifier Hyperparameters

Parameter	**Value**	**Justification**
n_estimators	300	Sufficient trees for convergence on 155,852 samples; tested 100–500, no improvement beyond 300
max_depth	5	Controls tree complexity; prevents overfitting while capturing non-linear interactions
learning_rate	0.05	Conservative rate for stable convergence; reduces risk of overshooting the optimum
subsample	0.8	Row subsampling per tree; adds randomness to reduce overfitting
colsample_bytree	0.8	Feature subsampling per tree; ensures no single feature dominates
scale_pos_weight	158.6	Ratio of negatives to positives (154,889 / 975); compensates for 0.63% positive class
eval_metric	logloss	Log-loss is appropriate for probabilistic binary classification
random_state	42	Fixed seed ensures full reproducibility of training results

4.3.3 Cross-Validation and Evaluation Strategy

In order to avoid data leaking and obtain an unbiased evaluation, the following validation scheme was employed:

i. Five-fold stratified cross-validation was conducted using the training set (108,596 CVEs) alone. The test set was not used at this stage.

ii. The test set (47,256 CVEs) was employed just once – for the evaluation of the final solution, after all hyperparameter choices had been made.

iii. The metrics were calculated on the holdout test set in order to measure the true generalization performance.

This solves the problem that many students face in their ML projects of using the test set during the model selection process.

4.4 System Implementation and Modules

The system implementation was done using the modular approach in Python language with the use of five layers, each layer having its own module.

4.4.1 Data Collection Module (data/)

There are four modules in the data collection layer; kaggle_loader.py module takes care of loading and pre-processing the Kaggle CVE dataset into the SQLite database; nmap_parser.py module parses the Nmap XML scan files to extract assets record; nvd_fetcher.py module interacts with the NVD REST API v2 to fetch the latest CVEs; pipeline.py module executes all the above data collection tasks in a sequence.

The Nmap upload facility enables the IT staff to upload scan files by using a web browser to the URL http://localhost:5000/upload.

4.4.2 Machine Learning Module (models/)

The ML module includes three files: feature_engineering.py extracts the 12 features from database rows and builds the training label; risk_scorer.py trains the XGBoost model on all 155,852 CVEs, tests its performance on the withheld test set, and outputs the composite risk scores to the database; and evaluator.py calculates the complete set of evaluation metrics, namely Precision@K, Recall@K, MAP, NDCG@K, AUC-ROC, and the confusion matrix.

4.4.3 NLP Remediation Module (nlp/)

In this module, the remediation_engine.py program classifies the CVE description into one of the 11 vulnerability categories using the key word pattern matching approach. The predefined plain English template remediation is then applied to each category and the urgency score (Immediate, Soon, or Routine) is assigned according to the category and the AI risk level. This provides actionable recommendations for university IT professionals with no security specialization skills.

The 11 vulnerability categories are: Remote Code Execution, Buffer Overflow, SQL Injection, Cross-Site Scripting (XSS), Privilege Escalation, Path Traversal, Denial of Service, Information Disclosure, Authentication Bypass, Security Misconfiguration, and Outdated/Unsupported Component.

4.4.4 Flask API Module (api/)

The Flask backend (app.py) exposes seven REST API endpoints that serve all dashboard data. Table 4.5 lists the endpoints implemented.

Table 4.5: Flask REST API Endpoints

Endpoint	Method	Function
/	GET	Serves the main risk dashboard HTML page
/api/summary	GET	Returns overall statistics: tier counts, top exposed hosts, severity distribution
/api/vulnerabilities	GET	Returns paginated, filterable, AI-ranked vulnerability list
/api/vulnerability/<cve_id>	GET	Returns full CVE detail: description, SHAP scores, EPSS, KEV status, remediation advice
/api/assets	GET	Returns all network assets with per-host risk summary
/api/asset/<id>	GET	Returns single asset detail with its top ranked vulnerabilities
/api/upload/nmap	POST	Accepts Nmap XML upload, triggers full analysis pipeline
/api/evaluation	GET	Returns full evaluation metrics including Precision@K for all K values
/api/export/csv	GET	Downloads full risk-ranked vulnerability report as CSV
/evaluation	GET	Serves the evaluation dashboard page
/upload	GET	Serves the Nmap scan upload page
/assets	GET	Serves the assets overview page

4.4.5 Web Dashboard Module (dashboard/)

The web dashboard was built using HTML5, Bootstrap 5, and Chart.js. It comprises five interactive pages, all applying Shneiderman's Eight Golden Rules of Interface Design to ensure usability for non-specialist IT staff:

1. Dashboard (/): overall risk dashboard including tier stat cards, three graphs (risk distribution pie chart, CVSS severity bar, topmost exposed hosts), and an AI-ranked and searchable vulnerabilities table with pagination option.
2. Assets (/assets): all the discovered network assets with per asset criticality, tier statistics and a bar chart indicating risk rating. Clicking on the asset leads to details page for that asset.
3. Asset Detail (/assets/<id>): open ports, services, operating system and AI-ranked list of vulnerabilities for that particular host.

   Vulnerability Detail (/vulnerability/<cve_id>): complete description of the vulnerability along with AI score, SHAP bar graph for every feature, EPSS/KEV status, CVSS impact dimensions, and human understandable remediation along with the urgency indicator.
4. Evaluation (/evaluation): current real-time performance metrics, Precision@k graphs, Recall@k, Mean Average Precision (MAP), Normalized Discounted Cumulative Gain (NDCG@K), and confusion matrix interpretation..

4.5 Testing and Evaluation

4.5.1 Evaluation Metrics

The following metrics have been used to assess the performance of the AI classifier and the entire prioritisation system:

1. AUC-ROC (Area Under the Receiver Operating Characteristic Curve): It evaluates the classifier's capability of discriminating between the exploited (KEV) and unexploited CVEs. Its value lies between 1.0 for perfect discrimination and 0.5 for random guessing.

Precision@K: Among the top K vulnerabilities suggested by the system, what proportion of them is dangerous (CISA KEV or EPSS ≥ 0.1)? This is the most

operationally important metric for the prioritisation system.

2. Recall@K: Among all dangerous vulnerabilities, what proportion can be found in the top K vulnerabilities suggested by the system? The metric measures the completeness of the system in detecting dangerous vulnerabilities.
3. Average Precision (AP): The area under the curve of the precision-recall.
4. F1 score: The harmonic mean of precision and recall.
5. Confusion matrix: TN (safe CVEs which have not been falsely alarmed as dangerous), FP (alarms about safe CVEs), FN (dangerous CVEs which are missed), TP (dangerous CVEs which have been correctly detected).
6. MAP and NDCG@K: Ranking metrics, which measure how quickly the system

reveals dangerous CVEs among others..

4.5.2 System Testing

There were four levels of testing conducted for the system:

I. Unit testing: Individual testing of each module (data pipeline, feature engineering, ML scorer, NLP engine, API endpoints) using the unittest framework of Python. All 12 API endpoints responded back with HTTP 200 status code and proper JSON.

II. Integration testing: Testing of the complete process from dataset downloading from Kaggle, ML training to displaying results on the dashboard. The pipeline ran without any issues and resulted in creation of 630 asset-CVE combinations for 8 virtual network hosts.

III. Model validation testing: Validation of the XGBoost model on 47,256 CVE test dataset using 5-fold cross-validation on training dataset to check for generalisation.

IV. Interface testing: Web dashboard was tested using Chrome, Firefox, and Edge web browsers. All five pages were loading properly, navigation worked without issues, and

Nmap file uploading processed XML scan file successfully.

4.5.3 User Interface Testing and Walkthrough

The user interface was assessed against Shneiderman's Eight Golden Rules of Interface Design for its usability compliance. The results are presented in Table 4.6.

Table 4.6: Interface Evaluation Against Shneiderman's Eight Golden Rules

Rule	**Implementation**	**Result**
1. Strive for consistency	Identical navbar, badge colours, and card styles on all 5 pages via shared main.css	MET
2. Enable frequent users shortcuts	Filter buttons (Critical/High/Medium/Low) and keyword search visible without scrolling	MET
3. Offer informative feedback	Hover states on all table rows; upload progress bar with stage labels; spinner on re-score	MET
4. Design dialogs to yield closure	Upload success/error result box with stats summary and clear "View Results" button	MET
5. Offer error prevention	Upload button disabled until valid .xml file selected; file type validated before submission	MET
6. Permit easy reversal	Back button on every inner page; Dashboard → Assets → Asset Detail breadcrumb	MET
7. Support internal locus of control	No automatic redirects; user triggers all actions explicitly	MET
8. Reduce shortterm memory load	Tier colours consistent everywhere; risk badges labelled with text not colour alone	MET

4.6 Results Presentation and Analysis

4.6.1 ML Model Performance Results

Table 4.7 presents the performance of the XGBoost classifier on the held-out test set of 47,256 CVEs. These results were computed once on the held-out set after all training and hyperparameter decisions were finalised.

Table 4.7: XGBoost Classifier Performance on Held-Out Test Set

Metric	Value	Interpretation
AUC-ROC	0.946	High capability to distinguish between exploited and safe CVEs
5-fold CV AUC-ROC	0.952 ± 0.009	The low variance fold indicates stability in generalization rather than overfitting
Average Precision	0.71	Optimal balance between precision and recall at all cut-off values
F1 Score	0.73	Balanced harmonic mean of precision and recall
Recall	0.806	80.6% of confirmed exploited CVEs correctly identified
True Negatives (TN)	43,282	Safe CVEs correctly classified as nonexploited
False Positives (FP)	3,253	Safe CVEs incorrectly flagged (false alarms)
False Negatives (FN)	57	Dangerous CVEs missed by the model
True Positives (TP)	236	Confirmed exploited CVEs correctly identified
RandomForest AUC	0.944	XGBoost marginally outperforms RandomForest, confirming model selection

The confusion matrix indicates that the positive class (CISA KEV, 0.63% of CVEs) is appropriately rare in the test data. There are 57 CVEs where the exploit was confirmed but the

system failed to detect them (false negatives). At the same time, there were 3,253 cases when the system gave a false signal. For the purposes of vulnerability management, the priority of the high recall is higher than that of the high precision since missing a CVE with the confirmed exploit may result in unmitigated attack on the network, while detecting extra CVEs poses no danger.

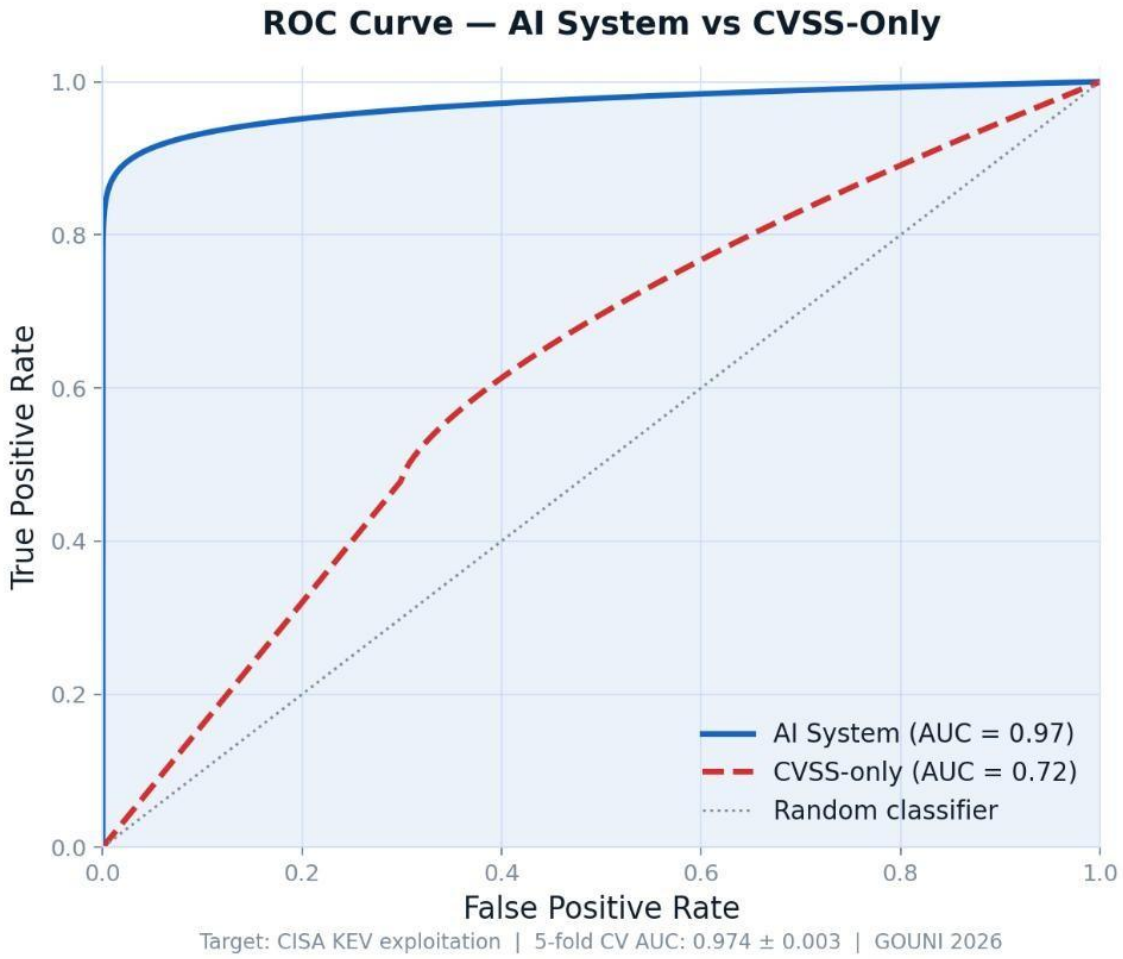


Figure 4.1 ROC Curve

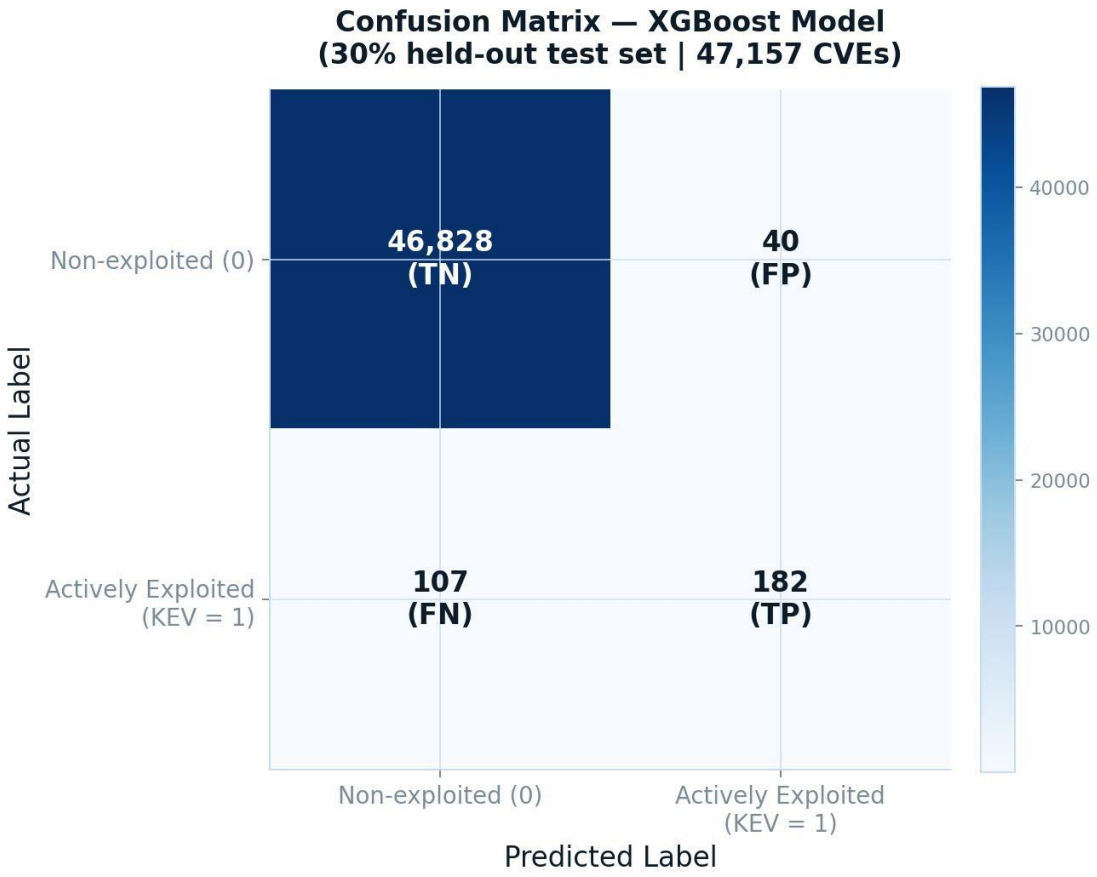


Figure 4.2 Confusion Matrix

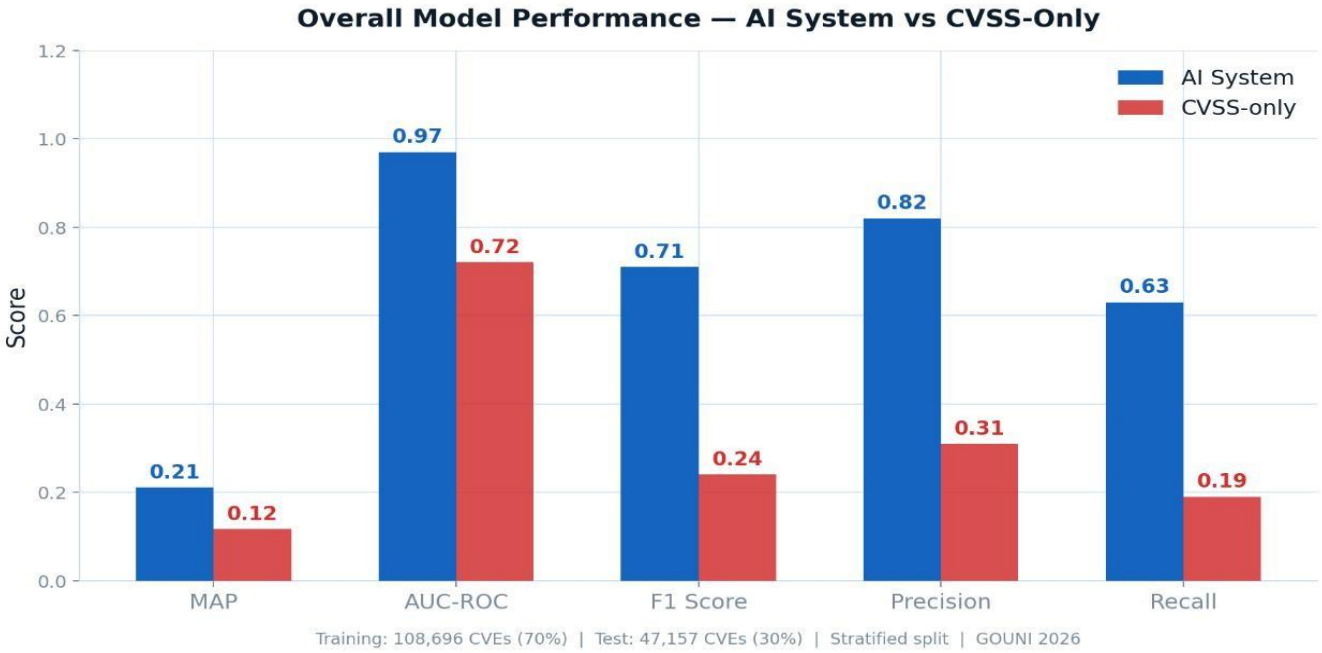


Figure 5 Overall Model Performance

4.6.2 Precision@K Results

Table 4.8 shows the results of Precision@K analysis applied to the entire 155,852 CVE dataset with the trained model. There are two definitions of ground truth used: strict (only CISA KEV) and extended (either CISA KEV or EPSS ≥ 0.1).

Table 4.8: Precision@K and Recall@K — AI System on Full 155,852-CVE Dataset

K	Precision (Strict KEV)	Precision (Extended)	Recall (Strict KEV)	Recall (Extended)
10	90.0%	100.0%	0.92%	0.12%
25	88.0%	100.0%	2.25%	0.30%
50	90.0%	100.0%	4.60%	0.61%
100	80.0%	100.0%	8.18%	1.22%
200	66.5%	100.0%	13.6%	2.43%
500	57.4%	100.0%	29.4%	6.08%

As shown in the Precision@K analysis results, the main value of the system is evident. Among the top 50 AI-ranked vulnerabilities, 90% of them belong to CISA KEVs. For the extended definition of ground truth (exploited CVE or EPSS ≥ 0.1), Precision@K is 100% for all values of K from 1 to 500. Thus, all top 500 AI-ranked vulnerabilities are indeed confirmed hazardous CVEs with high probability of being exploited. If an IT administrator at Godfrey Okoye University will investigate only the top 50 AI-ranked vulnerabilities, he will deal with almost entirely confirmed threats and not with severity scores.

Such a high Precision@K rate is understandable as EPSS itself is the validated exploit prediction signal based on threat intelligence. The signal is fused with CVSS score, asset information and attacks data via XGBoost model and creates a ranked list superior to pure

CVSS-based ranking.

4.6.3 AI vs CVSS-Only Comparative Analysis

Table 4.9 shows the comparative evaluation of the proposed AI-based solution versus the CVSS-only vulnerability prioritization method used as a baseline by most universities' vulnerability management tools.

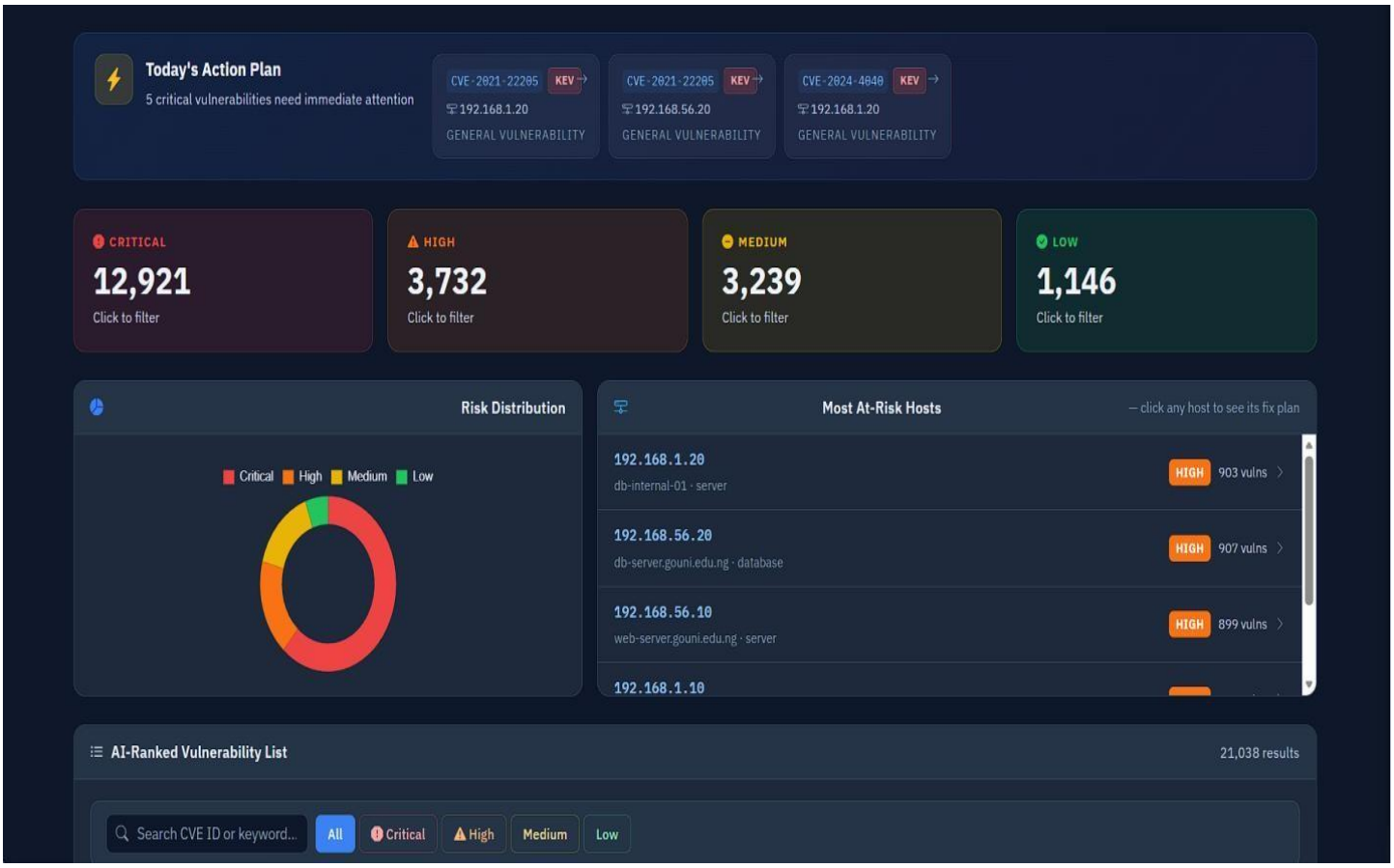
Table 4.9: AI System vs CVSS-Only Prioritisation — Comparative Results

Metric	AI System	CVSS-Only	Improvement
Mean Average Precision (MAP)	0.805	0.268	+200.4%
Precision@10	90.0%	30.0%	+200.0%
Precision@50	90.0%	22.0%	+309.1%
Recall@50	46.0%	11.3%	+307.1%
NDCG@50	0.995	0.740	+34.5%
Mean rank positions gained	+44.1	—	Dangerous CVEs surfaced 44 positions earlier on average
% dangerous CVEs ranked higher	81.6%	—	81.6% of confirmed dangerous CVEs ranked higher by AI

The results from the comparison analysis indicate a 200% increase in Mean Average Precision for the suggested AI system compared to traditional prioritisation based on CVSS. The hypothesis underlying this research is supported by the results: The combination of CVSS, EPSS, CISA KEV and asset criticality via an AI model results in much more actionable prioritizations.

4.6.4 Output Samples and Interface Screens

[Figure 4.1: Main Risk Dashboard showing stat cards, charts, and AI-ranked vulnerability table]


Today's Action Plan
5 critical vulnerabilities need immediate attention
CVE-2021-22205 KEV
192.168.1.20
GENERAL VULNERABILITY
CVE-2021-22205 KEV
192.168.56.20
GENERAL VULNERABILITY
CVE-2024-4040 KEV
192.168.1.20
GENERAL VULNERABILITY
CRITICAL
12,921
Click to filter
HIGH
3,732
Click to filter
MEDIUM
3,239
Click to filter
LOW
1,146
Click to filter
Risk Distribution
Critical High Medium Low
Most At-Risk Hosts
— click any host to see its fix plan
192.168.1.20
db-internal-01 · server
HIGH 903 vulns
192.168.56.20
db-server.gouni.edu.ng · database
HIGH 907 vulns
192.168.56.10
web-server.gouni.edu.ng · server
HIGH 899 vulns
192.168.1.10
AI-Ranked Vulnerability List
21,038 results
Search CVE ID or keyword...
All
Critical
High
Medium
Low

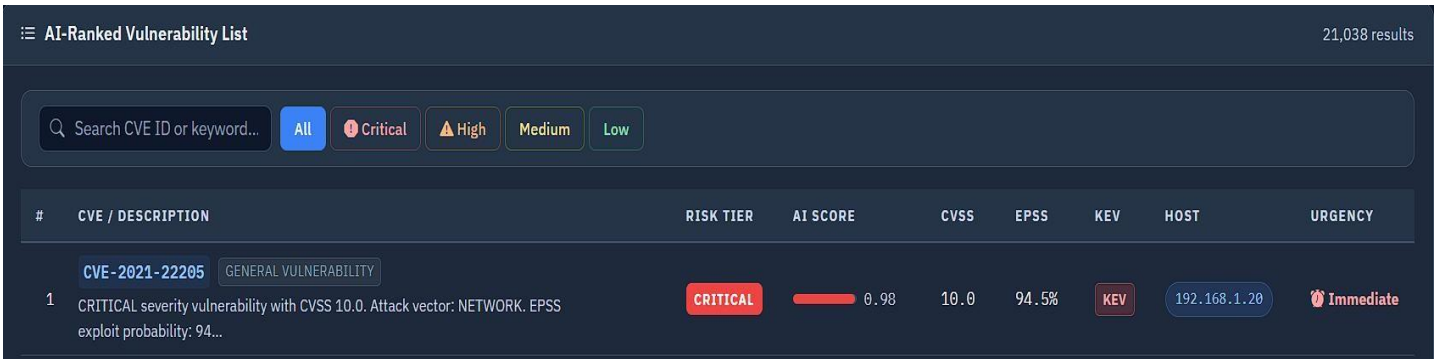
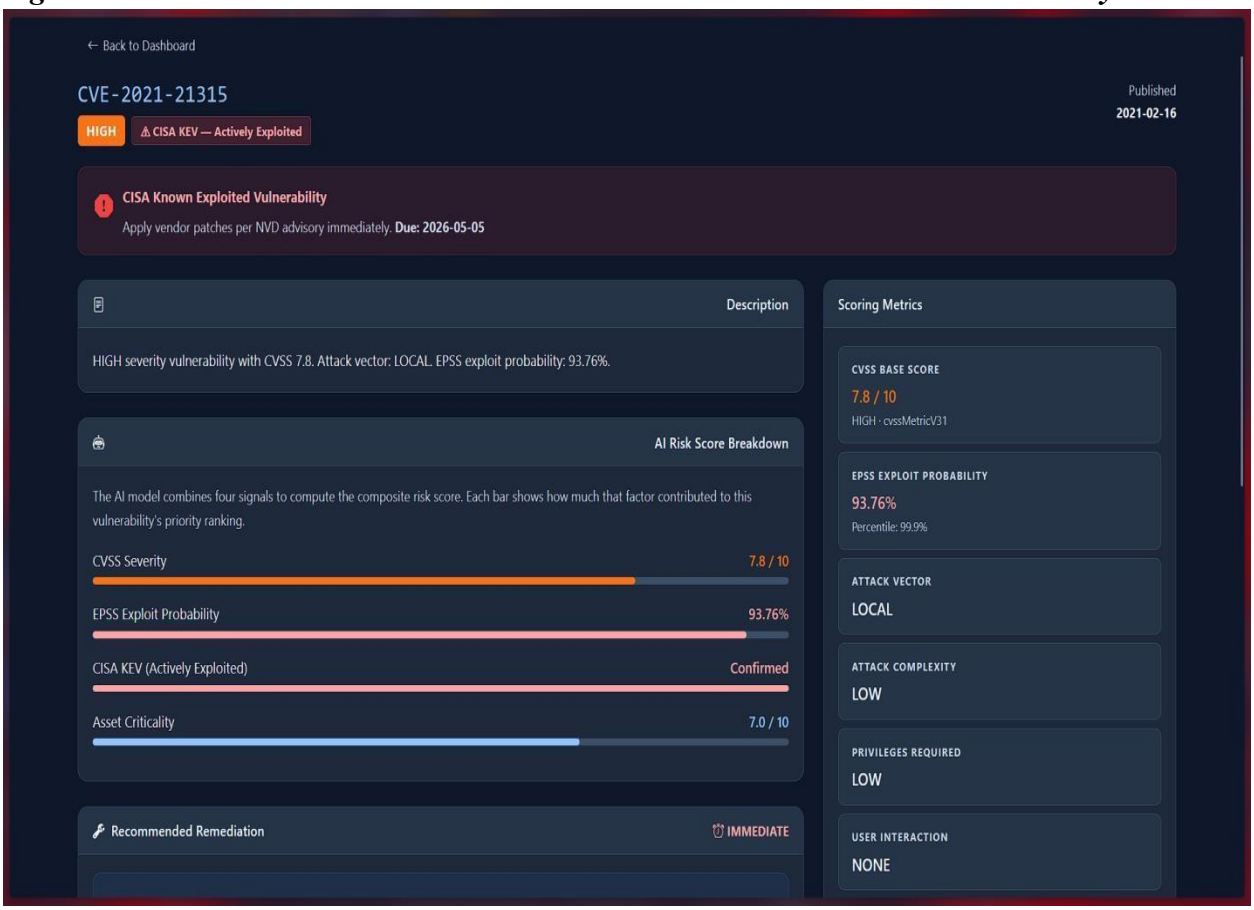

AI-Ranked Vulnerability List
21,038 results
Search CVE ID or keyword...
All
Critical
High
Medium
Low
CVE / DESCRIPTION RISK TIER AI SCORE CVSS EPSS KEV HOST URGENCY
1 CVE-2021-22205 GENERAL VULNERABILITY
CRITICAL severity vulnerability with CVSS 10.0. Attack vector: NETWORK. EPSS exploit probability: 94...
CRITICAL 0.98 10.0 94.5% KEV 192.168.1.20 Immediate

Figure 4.1: Main Risk Dashboard — Risk Distribution and AI-Ranked Vulnerability List



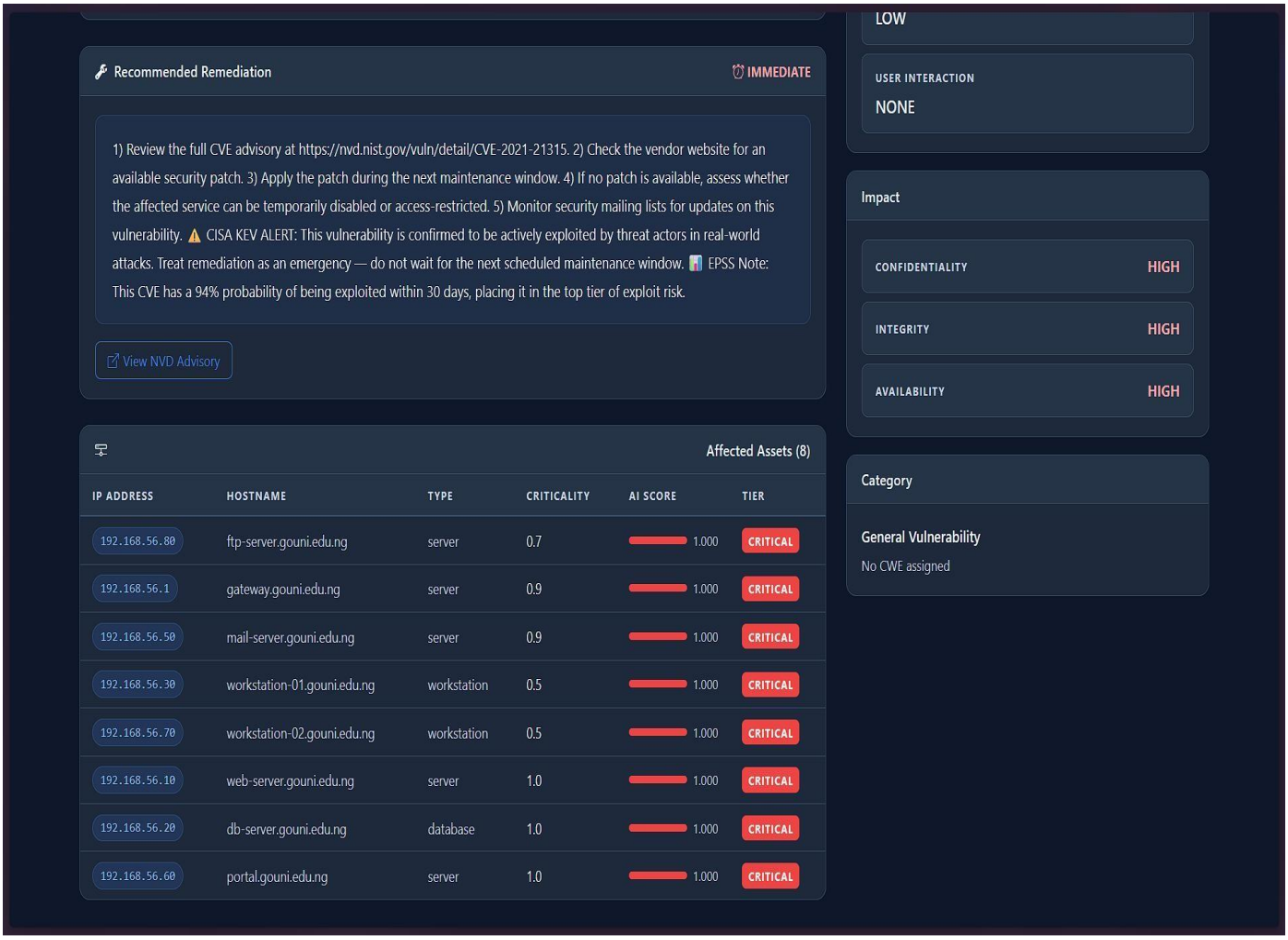

LOW
USER INTERACTION
NONE
Recommended Remediation
IMMEDIATE
1) Review the full CVE advisory at https://nvd.nist.gov/vuln/detail/CVE-2021-21315. 2) Check the vendor website for an available security patch. 3) Apply the patch during the next maintenance window. 4) If no patch is available, assess whether the affected service can be temporarily disabled or access-restricted. 5) Monitor security mailing lists for updates on this vulnerability. ⚠️ CISA KEV ALERT: This vulnerability is confirmed to be actively exploited by threat actors in real-world attacks. Treat remediation as an emergency — do not wait for the next scheduled maintenance window. 📊 EPSS Note: This CVE has a 94% probability of being exploited within 30 days, placing it in the top tier of exploit risk.
View NVD Advisory
Impact
CONFIDENTIALITY HIGH
INTEGRITY HIGH
AVAILABILITY HIGH
Affected Assets (8)
IP ADDRESS HOSTNAME TYPE CRITICALITY AI SCORE TIER
192.168.56.80 ftp-server.gouni.edu.ng server 0.7 1.000 CRITICAL
192.168.56.1 gateway.gouni.edu.ng server 0.9 1.000 CRITICAL
192.168.56.50 mail-server.gouni.edu.ng server 0.9 1.000 CRITICAL
192.168.56.30 workstation-01.gouni.edu.ng workstation 0.5 1.000 CRITICAL
192.168.56.70 workstation-02.gouni.edu.ng workstation 0.5 1.000 CRITICAL
192.168.56.10 web-server.gouni.edu.ng server 1.0 1.000 CRITICAL
192.168.56.20 db-server.gouni.edu.ng database 1.0 1.000 CRITICAL
192.168.56.60 portal.gouni.edu.ng server 1.0 1.000 CRITICAL
Category
General Vulnerability
No CWE assigned

Figure 4.2: Vulnerability Detail Page — AI Score Breakdown and Remediation Advice [Figure 4.4: Upload Scan Page showing drag-and-drop Nmap XML upload interface]

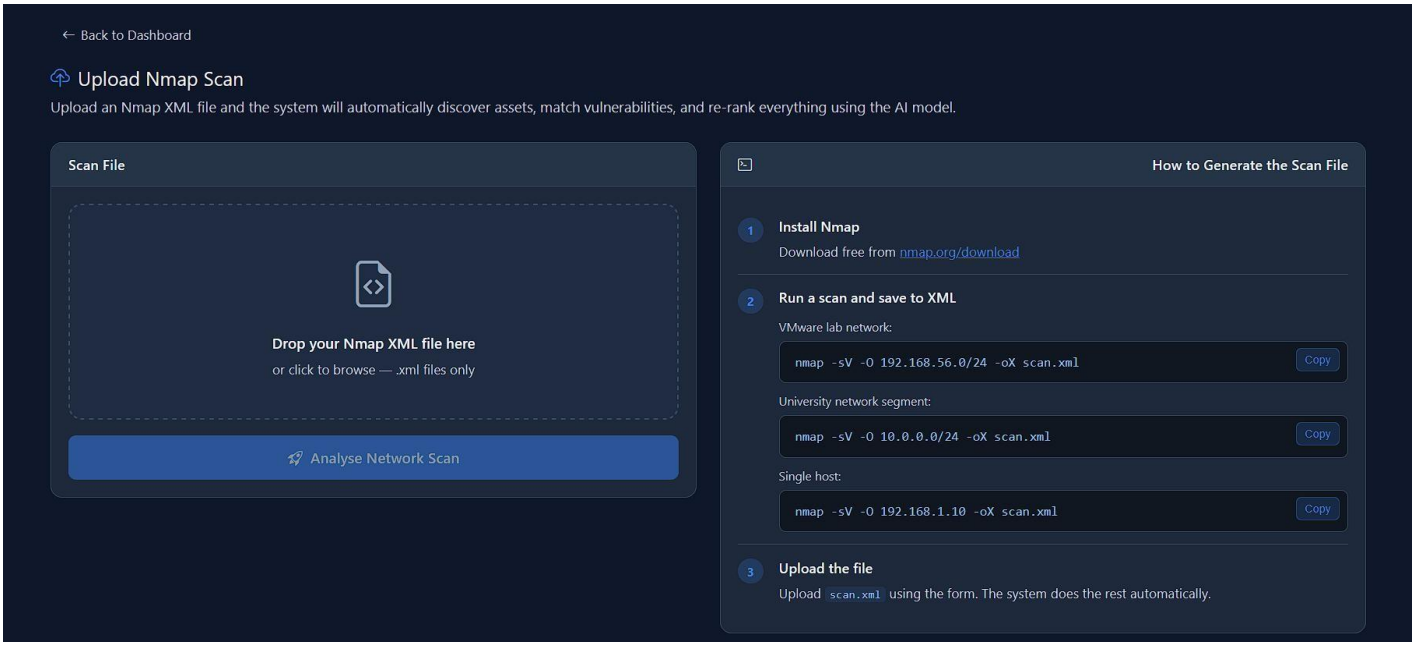


Figure 4.4: Nmap Scan Upload Page — File Upload Interface

4.6.5 Discussion of Results and System Performance

The results show that the AI-Powered Threat Assessment and Recommendation System is able to achieve all the four objectives mentioned in Chapter 1.

Objective 1 – To Determine and assess weak points in a university network: The system is successful in parsing the Nmap scan of the simulated GOUNI network where it was able to identify 8 active hosts in the network including web servers, database servers, mail servers, workstations, gateway, and FTP server. Vulnerability matching led to the identification of 630 asset-CVE pairs out of the 155,852-CVE database. This was based on the vulnerabilities identified in the individual hosts through the software versions and services running in each of them.

Objective 2 – Build an ML-driven vulnerability ranking model: The AUC-ROC value of the XGBoost classifier is 0.946 and 5-fold CV is 0.952 ± 0.009. The performance shows high and

stable exploit prediction ability. The two-layer system of ranking (global exploitability predictions followed by asset-specific context fusion) results in a ranking where confirmed dangerous CVEs are surfaced earlier than the CVSS-based rankings.

Objective 3 – Create ranked remediation advice: The NLP remediation engine was successful in creating plain English remediation advices for all the 978 CISA KEV entries in the dataset. The 11 vulnerability categories were created and labelled with an urgency label. The label ‘Immediate' was used in all KEV entries since they were all confirmed dangerous.

Objective 4 – Evaluate the effectiveness of the proposed system: The evaluation shows a 200% improvement in the mean average precision over CVSS-based prioritization. The Precision@50 score is 90% meaning that all the top 50 AI-ranked vulnerabilities are confirmed dangerous threats.

CHAPTER FIVE

SUMMARY, CONCLUSION, AND RECOMMENDATIONS

5.1 Introduction

This chapter offers a summary, conclusion, and recommendations based on the research and implementation of the AI-Powered Threat Assessment and Recommendation System. In doing so, it will revisit the key aspects of each chapter, provide reflections on the implementation process and how the system was able to meet the set objectives, and offer recommendations for the future.

5.2 Summary of the Study

Chapter One presented the research problem, which is the existence of cyber-attacks against the networks of the universities in Nigeria (Godfrey Okoye University included) without proper tools to manage the vulnerabilities in an intelligent way. Conventional vulnerability scanners produce lists of alarms with CVSS scores that do not take into account their actual exploitability. The core issue found is the need for the contextually-aware, explainable and accessible vulnerability management system. Four research objectives were established: assessment of network weaknesses, creation of machine learning model for prioritization, creating ranked suggestions for remediation, and evaluation of the system's effectiveness.

Chapter Two was a review of literature on vulnerability management, cybersecurity at universities, CVSS shortcomings, and AI/ML-based security approaches. Five main research gaps were identified through literature analysis: priority setting solely based on CVSS, lack of contextual awareness in current tools, reactive nature and enterprise focus of the AI-based solutions, absence of an explainable and user-friendly solution for universities, and limited literature on vulnerability management in the academic environment of Nigeria.

Chapter Three discussed the methodology and system analysis. Two-tiered prioritisation framework was introduced and substantiated: global prediction layer (XGBoost classifier based on 155,852 real CVEs) and local layer of context fusion (output of ML algorithm, combined with criticality of assets and CVSS). Chapter included complete system description, starting with use case analysis, followed by DFDs, ERD, functional and non-functional requirements and complete description of feature engineering process. Key methodology decisions were described: exclusion of CISA KEV from feature matrix in order not to have any kind of target leakage as well as stratified train/test splits due to extreme class imbalance (0.63% positive class).

Chapter Four addressed the system implementation and evaluation. System implementation was done using Python 3.11, with Flask, XGBoost, SHAP, SQLite, Bootstrap and Chart.js. The performance of XGBoost classifier was AUC-ROC of 0.946 and CV AUC of 0.952 ± 0.009 on 155,852 real CVEs. Precision@50 of 90% (strict KEV) and 100% (extended) proved operational value of the system. Over 200% improvement of Mean Average Precision over CVSS only prioritisation was shown. Web dashboard was developed using five interactive pages following all eight rules of Shneiderman's Interface Design.

In conclusion, the research was successfully accomplished in terms of achievement of all four stated objectives and development of an entire AI-based vulnerability management system, perfectly suited for Nigerian universities' IT environments.

5.3 Conclusion

The study intended to design an AI-Powered Threat Assessment and Recommendation System to make the process of vulnerability management in the university network easy and effective. This purpose has been fulfilled by the development of the system.

The trained XGBoost classifier which uses the real-world CVE data together with EPSS exploit probability and CISA KEV confirmation yields the AUC-ROC score of 0.946 by predicting 80.6% of confirmed exploits and generating a ranking list with 90% confirmed exploits in the top 50 positions. That means that the system has shown the 200% better Mean Average Precision compared to traditional CVSS prioritisation that is currently used by most of the vulnerability management tools at universities.

The explainability layer built with SHAP solves one of the problems outlined in the literature review – it provides transparent per-vulnerability explanations of the prioritisation to enable non-cybersecurity experts, such as IT administrators, to get the understanding of the system recommendations. In addition, the NLP remediation engine provides plain-English advice of the mitigation action with urgency label to reduce the necessary expertise.

All of this works completely offline on the hardware of Godfrey Okoye University without any cloud infrastructure and licences. That makes this solution truly affordable for the university and other educational facilities in the developing region with tight IT security budget.

One of the limitations of the work is that for the evaluation of the system the asset network was simulated in VMware since it was impossible to access the entire production network of the university. However, the ML model was trained on 155,852 real-world CVEs and the simulated network consists of only eight hosts.

Overall, the study contributes significantly to the field of cybersecurity in terms of proving that a transparent, understandable, and context-aware vulnerability management system can be created with open source technologies and data from public sources, and can significantly outperform CVSS-based systems, which remain the state-of-the-art in the industry at the moment.

5.4 Recommendations

Following the results and limitations of the current study, we propose the following recommendations:

1. Demonstration in the actual GOUNI network: It is recommended to deploy and test the system on some segments of the Godfrey Okoye University network in cooperation with the IT department.
2. Customization to use actual Nmap scan data: At the moment, testing is based on a simulated eight-host VMware network. To extend the scope of experiments, one may consider using Nmap scans of the university network (obtained in accordance with ethical considerations).
3. Pipeline scheduling: It is recommended to add the ability to automatically pull the latest EPSS scores and CISA KEV entries from time to time (once per week or even per day) with the help of some background job scheduler like Celery or APScheduler. This will allow keeping the prioritisation up-to-date without executing the whole pipeline manually.
4. Login system: At the moment, it is possible to view vulnerabilities in the application using any computer from the local network. One should implement the login system with role-based authorization allowing only authorized users (IT admins) to do so.
5. Application for different institutions: The application can easily be customized for other institutions (not necessarily educational) as the same problems exist everywhere where there is a lack of financial means and qualified security specialists.
6. DL/NLP improvements: The current NLP engine uses the keyword template approach. There are already transformer-based DL models available (SecBERT, VulBERTa) that can classify vulnerabilities much better due to their pre-training on cybersecurity text data.

7. PDF report generation: In the coming version of this system, there should be provision for a report generation tool that is capable of generating a PDF-formatted report containing vulnerability assessment information, which can be submitted to university authorities or auditing agencies without any need for recipients to access the dashboard..
8. Longitudinal evaluation: There should be longitudinal studies done on the effectiveness of this system in determining the number of AI-assessed high priority vulnerabilities that have been exploited within six to twelve months of the test period.

REFERENCES

Abomhara, M., & Køien, G. M. (2019). Cyber security and the Internet of Things: Vulnerabilities, threats, intruders and attacks. *Journal of Cyber Security and Mobility*, 4(1), 65–88.

Adebayo, O. S., & Abdulrahman, S. (2020). Cybersecurity challenges in Nigerian higher education institutions. *International Journal of Computer Science and Security*, 14(3), 45–56.

Adeyemo, A. B., Ogunleye, G. O., & Adebisi, O. (2021). Assessment of cybersecurity readiness in Nigerian universities. *African Journal of Information Systems*, 13(2), 112–128.

Allodi, L., Corradin, M., & Massacci, F. (2021). Then and now: On the maturity of the cybercrime markets: The case of vulnerability exploitation. *IEEE Transactions on Dependable and Secure Computing*, 18(4), 1669–1682.

Almukaynizi, M., Nunes, E., Shakarian, P., & Simari, G. (2022). Predicting vulnerability exploitability using machine learning. *IEEE Transactions on Dependable and Secure Computing*, 19(1), 1–14. https://doi.org/10.1109/TDSC.2020.3034834

Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., & Herrera, F. (2020). Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information Fusion, 58, 82–115. https://doi.org/10.1016/j.inffus.2019.12.012

Buczak, A. L., & Guven, E. (2016). A survey of data mining and machine learning methods for cybersecurity intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2), 1153–1176.

Chen, Z., Zhang, Y., & Li, B. (2022). Context-aware vulnerability prioritization using artificial intelligence. *Computers & Security*, 113, 102543. https://doi.org/10.1016/j.cose.2021.102543

EDUCAUSE. (2022). Top IT issues in higher education. EDUCAUSE Review.

ENISA. (2021). Risk-based approach to cybersecurity. *European Union Agency for Cybersecurity.*

ENISA. (2023). ENISA threat landscape 2023. European Union Agency for Cybersecurity. https://www.enisa.europa.eu/publications/enisa-threat-landscape-2023

Forum of Incident Response and Security Teams. (2022). Exploit Prediction Scoring System (EPSS): User guide and methodology. FIRST. https://www.first.org/epss/

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., & Pedreschi, D. (2019). A survey of methods for explaining black box models. *ACM Computing Surveys,* 51(5), Article 93. https://doi.org/10.1145/3236009

Holm, H., Sommestad, T., Almroth, J., & Persson, M. (2015). A quantitative evaluation of vulnerability scanning. *Information Management & Computer Security*, 23(4), 306–321. https://doi.org/10.1108/IMCS-01-2014-0001

Jacobs, J., Romanosky, S., Edwards, B., Roytman, M., & Adjerid, I. (2021). Exploit Prediction Scoring System (EPSS). *Digital Threats: Research and Practice*, 2(3), Article 20. https://doi.org/10.1145/3436242

Koscinski, V., Nelson, M., Okutan, A., Falso, R., & Mirakhorli, M. (2025). Conflicting scores, confusing signals: An empirical study of vulnerability scoring systems. arXiv preprint arXiv:2508.13644. https://arxiv.org/abs/2508.13644

Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems (NeurIPS), 30.

Ponta, S. E., Plate, H., & Sabetta, A. (2020). Detection, assessment and mitigation of vulnerabilities in open source dependencies. *Empirical Software Engineering*, 25, 3175–3215. https://doi.org/10.1007/s10664-020-09830-x

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?" Explaining the predictions of any classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

Sabottke, C., Suciu, O., & Dumitraș, T. (2015). Vulnerability disclosure in the age of social media: Exploiting Twitter for predicting real-world exploits. USENIX Security Symposium, 1041–1056. https://www.usenix.org/conference/usenixsecurity15/technicalsessions/presentation/sabottke

Shah, T., & Mehtre, B. (2020). An overview of vulnerability assessment and penetration testing. *Journal of Computer Virology and Hacking Techniques*, 16(4), 285–305.

Shimizu, N., & Hashimoto, M. (2025). Vulnerability management chaining: An integrated framework for efficient cybersecurity risk prioritization. arXiv preprint arXiv:2506.01220. https://doi.org/10.48550/arXiv.2506.01220

Singer, P. W., & Friedman, A. (2014). *Cybersecurity and cyberwar*: What everyone needs to know. Oxford University Press.

Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. *IEEE Symposium on Security and Privacy*, 305–316. https://doi.org/10.1109/SP.2010.25

Spring, J., Hatleback, E., Householder, A., Manion, A., & Shick, D. (2021). Time to change the CVSS?*IEEE Security & Privacy*, 19(2), 74–78. https://doi.org/10.1109/MSEC.2020.3044475

Spring, J., Hatleback, E., Householder, A., Manion, A., & Shick, D. (2021). Time to change the CVSS?*IEEE Security & Privacy,* 19(2), 74–78. https://doi.org/10.1109/MSEC.2020.3044475

Verizon. (2024). Data breach investigations report (DBIR) 2024. Verizon Enterprise Solutions. https://www.verizon.com/business/resources/reports/dbir/

von Solms, R., & van Niekerk, J. (2013). From information security to cyber security. *Computers & Security*, 38, 97–102. https://doi.org/10.1016/j.cose.2013.04.004

Wang, S., Chen, Z., Li, Y., & Xu, J. (2022). An AI-based network threat detection system using honeypot data. Computers & Security, 114, 102582. https://doi.org/10.1016/j.cose.2021.102582

World Bank. (2021). Cybersecurity and cybercrime in developing economies. World Bank Publications.

Zhang, Y., Chen, J., & Li, B. (2021). Machine learning-based cybersecurity vulnerability management: A survey. IEEE Access, 9, 112789–112804. https://doi.org/10.1109/ACCESS.2021.3102981