A REPORT
ON

Web Scraping Tool for Critical & High-Severity OEM Vulnerabilities

Submitted by,

Shesha Venkat Gopal K	**20211CCS0053**
Chandrashekhar S	**20211CCS0065**
Shubha K A	**20211CCS0067**
Augustian P B	**20211CCS0104**

Under the guidance of,

Dr. Nihar Ranjan Nayak

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING [CYBERSECURITY]

At

PRESIDENCY UNIVERSITY

BENGALURU

MAY 2025

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Project report **"Web Scraping Tool for Critical & High-Severity OEM Vulnerabilities"** being submitted by Shesha Venkat Gopal K, Chandrashekhar S, Shubha K A, Augustian P B bearing roll number 20211CCS0053, 20211CCS0065, 20211CCS0067, 20211CCS0104 in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering [Cybersecurity] is a bonafide work carried out under my supervision.

Dr. NIHAR RANJAN NAYAK
Assistant Professor - Senior Scale
PSCS
Presidency University

Dr. ANANDARAJ S P
HoD
PSCS
Presidency University

Dr. MYDHILI NAIR
Associate Dean
PSCS
Presidency University

Dr. SAMEERUDDIN KHAN
Pro-Vice Chancellor - Engineering
Dean –PSCS / PSIS
Presidency University

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

I hereby declare that the work, which is being presented in the report entitled **"Web Scraping Tool for Critical & High-Severity OEM Vulnerabilities"** in partial fulfillment for the award of Degree of **Bachelor of Technology** in **Computer Science and Engineering [Cybersecurity]**, is a record of my own investigations carried under the guidance of **Dr. NIHAR RANJAN NAYAK, Assistant Professor - Senior Scale, Presidency School of Computer Science and Engineering, Presidency University, Bengaluru.**

I have not submitted the matter presented in this report anywhere for the award of any other Degree.

STUDENT NAME	ROLL NO	SIGNATURE
SHESHA VENKAT GOPAL K	20211CCS0053	
CHANDRASHEKAR S	20211CCS0065	
SHUBHA K A	20211CCS0067	
AUGUSTIAN P B	20211CCS0104	

ABSTRACT

Today digital business operations are very much dependent on hardware and software products provided by original equipment manufacturers (OEMs). Systems that are employed in operations expose security flaws that will be exploited by unauthorized parties in initiating attacks. The frequent distribution of advisory from OEMs creates a complex situation for these security teams since they require divergent update protocols for every platform. The manual operation of advisories causes inefficiencies in the system that can no longer be tolerated as advisory volumes occur over larger operations.

To fill this critical gap, this project aims to rollout an automated and intelligent system entitled "Web Scraping Tool for Critical & High-Severity OEM Vulnerabilities". The system utilizes automated monitoring to draw meaningful vulnerability information from OEM websites and segments severities and notifies recipients in real time. The following components are in place in this system; web scraping engines, with support from Natural Language Processing (NLP) and rule-based classification logic as well as a customizable notification framework.

A modular system runs core modules to extract static and dynamic website content data functions before executing concept and CVSS metric scanning tests and alarm notifications utilizing the secure email platforms. With such user friendly interface users maintain their sources and receive alerts during the process of report creation. The tool allows elegant extraction ethically since it follows terms of service regulations and standards that govern data privacy.

This tool allows organizations to move their vulnerability management from a delayed reaction to an early reaction by streamlining their responses towards critical advisories. The system allows the organizations to go beyond reactive defense by facilitating implementable risk based advisory classification for the purpose of vulnerability management.

The proposed system assists organisations in addressing their critical cybersecurity challenges by applying an intelligent legal-compliant approach; which makes digital infrastructure protection possible at high speed and accuracy.

ACKNOWLEDGEMENTS

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Pro-VC - Engineering and Dean, Presiency School of Computer Science and Engineering & Presiency School of Information Science, Presidency University for getting us permission to undergo the project.

We express our heartfelt gratitude to our beloved Associate Dean **Dr. Mydhili Nair,** Presidency School of Computer Science and Engineering, Presidency University, and **Dr. Anandaraj S P**, Head of the Department, Presidency School of Computer Science and Engineering, Presidency University, for rendering timely help in completing this project successfully.

We are greatly indebted to our guide **Dr. Nihar Ranjan Nayak, Assistant Professor - Senior Scale** and Reviewer **Dr. Shanthi S**, **Associate Professor**, Presidency School of Computer Science and Engineering, Presidency University for their inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the internship work.

We would like to convey our gratitude and heartfelt thanks to the PIP4004 University Project Coordinator **Mr. Md Ziaur Rahman and Dr. Sampath A K,** department Project Coordinators **Dr. Sharmasth Vali Y** and Git hub coordinator **Mr. Muthuraj.**

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

Shesha Venkat Gopal K

Chandrashekar S

Shubha K A

Augustian P B

LIST OF TABLES

LIST OF FIGURES

TABLE OF CONTENTS

Chapter 1
INTRODUCTION

1.1 Background

IT systems security remains an important bedrock upon which operations between public sector services and private sector businesses remain stable and resilient during this digital-first era. The modern technology world depends heavily on Original Equipment Manufacturer (OEM) products in constructing systems that integrate varied software elements with hardware devices ranging from enterprise applications – industrial controls to medical equipment and cloud platforms. Microsoft in collaboration with Cisco and Siemens and Adobe issue continuous security advisories which have step-by-step tutorials on how to mitigate and fix vulnerabilities found in their products.

The sharp increase in annual vulnerability disclosures is outpacing the ability of monitoring method to continue tracking consistently. The number of global Common Vulnerabilities and Exposures (CVE's) registered in 2023 exceeded 25,000 which represents the highest annual figure ever. The overwhelming number of vendor updates requires security teams to keep track of a growing number of advisory sources and analyze their relevance, and create appropriate responses diversity in the form of update variants.

Figure 1.1 displays the continuous growth of vulnerability reporting from 2020 through 2023 across high risk sectors including IT, finance, healthcare, and energy. IT, finance, healthcare, and energy. The IT sector continues to face the most cyber attacks because it spans extensive platforms while its critical systems require advanced network structures. Constant threats affect the financial sector since data and transactions represent valuable assets. Threat actors have intensified their focus on healthcare and energy infrastructure due to both sectors' essential nature combined with severe risks of service interruptions.

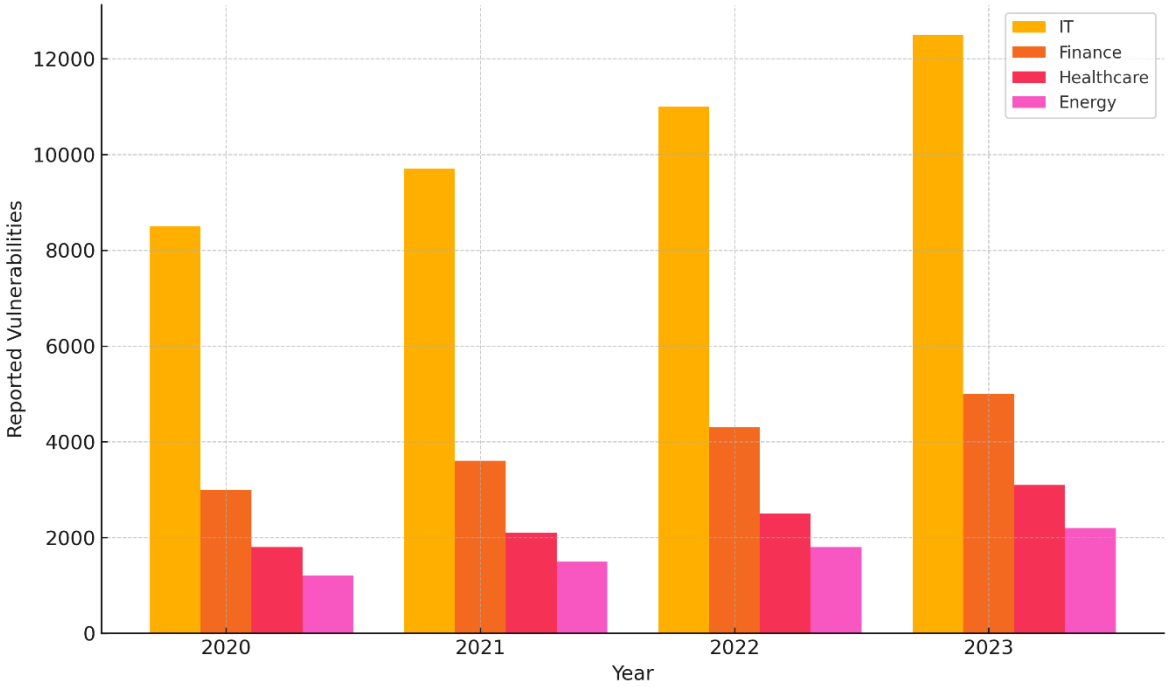


Figure 1.1: Global Cybersecurity Threat Landscape (2020–2023)

The accelerating number of security operations demands immediate changes to current defensive practices. Organizations need to improve their near-time capability to detect and respond to emerging security threats although vulnerabilities will continue to appear.

1.2 Motivation

The objective of this project is informed by an urgent need for improvement in vulnerability monitoring and management practices. Modern arrangements trace vulnerabilities in terms of manual process in which there is an effort to visit OEM websites and analyse and extract technical information which through impact assessment, it determines appropriate response. In turn users commence from OEM portals and advisory reading to go through data extraction and impact evaluation and response selection. This time-consuming approach to vulnerability management creates both operational inefficiency and safety risks.

Adverse high pressure situations arise out of a single missed advisory, and such situations can put systems at risk from known exploits. Correctly timed awareness about publication details tells one whether an attack is prevented or not. Notwithstanding, the exposure window that X-DNS is provided due to vulnerability disclosure, wherein attackers quickly convert these weaknesses into attack tools that same night within hours.

The automatic process used to detect OEM vulnerabilities as well as a classification along with alert generation allows organizations to reduce their security risks considerably. Security teams should move further from reactive practice to proactive threat management that will enable them to tackle risks before the adversaries.

1.3 Problem Statement

There are many hurdles that emanate when organizations attempt to manage vulnerabilities that come from original equipment manufacturers:

- **Non-standardized advisory formats**: OEMs create advisory content through three different document formats (HTML, JSON, PDF) using diverse terminology which makes automated parsing difficult.
- **Manual data extraction**: Data analysts should read numerous text formats to be able to extract important information such as CVE IDs and severity scores' among other affected components lists.
- **Delayed response times**: When automated processes are not used critical security alerts fail to deliver important information to the correct teams on schedule.
- **Alert fatigue**: The excessive number of low-priority alerts overloads team members who then fail to concentrate on severe exploitations.
- **Lack of integration**: Current tools show limited capability to efficiently collect and normalize vulnerability information from multiple vendor sources.

Some preventable attacks survive because the delay problems prevent organizations from carrying out good patch management.

1.4 Aim and Scope of the Project

This project provides a proposal for establishing an internet platform known as the "Web Scraping Tool for Critical and High-Severity OEM Vulnerabilities" with the objective to have automated vulnerability detection coupled to vulnerability categorization and vulnerability notification functionalities for OEM publications.

The project addresses these important objectives within their boundaries:

- **Automated Crawling**: Your application must maintain ongoing surveillance of OEM advisory portals to detect new content along with any updates.
- **Data Extraction**: The system uses sophisticated web scraping abilities with headless browsing protocols to retrieve vulnerability data even from pages that use dynamic content or JavaScript rendering.
- **Severity Classification**: Use rule-based approaches together with computer-assisted systems for determining advisory criticality levels.
- **Real-Time Alerts**: Stakeholders receive actionable notifications which trigger from customizable severity thresholds.
- **Dashboard Interface**: A user-friendly interface enables visual access to vulnerabilities and data management alongside report generation capabilities.
- **Data Storage**: Auditing and compliance requirements and trend analysis needs can be met through an archival system for advisories.

The security solution means that scalability is also extended alongside the extensibility, and ethical compliance characteristics to fit the current security infrastructure needs of medium to large organizations

1.5 Significance of the Project

The research examines how to solve one of the fundamental issues of cybersecurity: The system automates procedures for vulnerability monitoring of environments that require automated procedures rather than human-led process implementation. The system strives to enhance detection and classification effectiveness in vulnerabilities which has a direct impact on two key metrics of the incident response. Mean Time to Detect (MTTD) and Mean Time to Respond (MTTR).

With its implementation, the system provides organizations with an opportunity to monitor their risk exposures actively. Organizations are rewarded with improved security due to automated vulnerability assessment that is geared towards the reduction of human errors but also with high priority threat prioritizing and full operational tool integration including SIEM and email gateways. By delivering the real-time vulnerability detection the system enables the organizations to keep the GDPR and ISO 27001, as well as the NIST SP 800-82 framework compliant.

1.6 Key Challenges Addressed

In designing this system, several challenges must be overcome:

- **Data inconsistency** across OEM websites demands flexible scraping and parsing logic.
- **JavaScript-rendered content** necessitates the use of headless browsers like Selenium or Puppeteer.
- **Accurate classification** requires context-aware models that understand both keywords and CVSS scoring.
- **Alert customization** is needed to minimize noise and ensure the right people get the right information.
- **Ethical scraping practices** must be observed to avoid violating website terms or legal guidelines.

By addressing these challenges, the system offers a resilient solution that adapts to real-world constraints and organizational requirements.

1.7 The Cost of Delay: Understanding the Exploitation Window

The interval between revealing security holes and fixing them represents the main weakness in modern cybersecurity defense. Organizations including CISA and Tenable document that cyber attackers start taking advantage of disclosed vulnerabilities within 24 to 48 hours before organizations widely deploy security fixes.

Figure 1.2 depicts a standard timeline for vulnerability exploitation. Research indicates that an organization's vulnerability to attack grows intensely following public disclosure about a vulnerability while reaching its peak before most entities deploy the required updates.

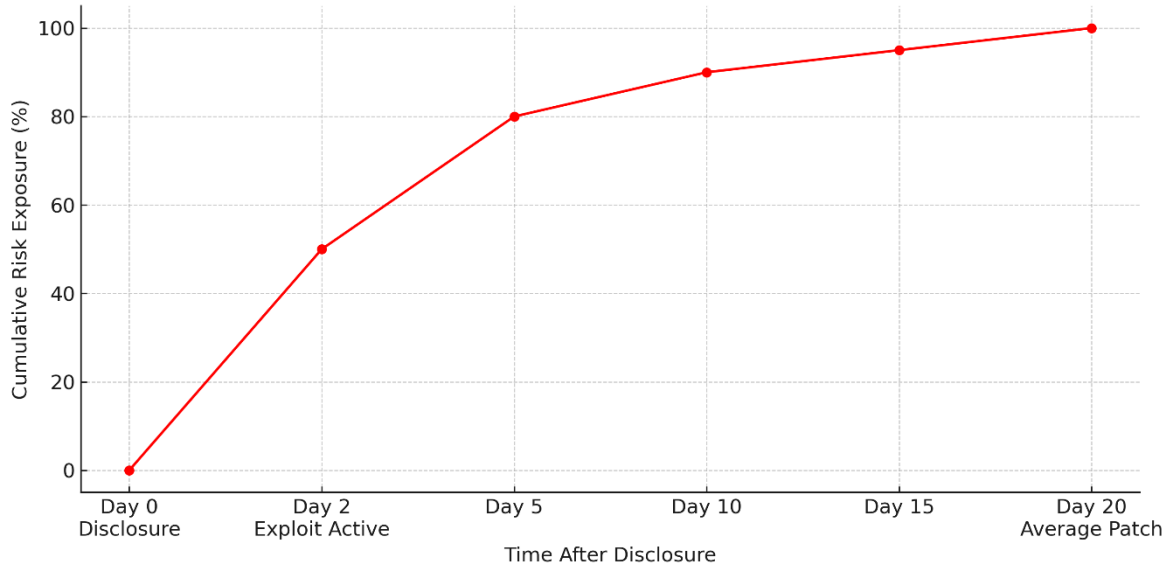


Figure 1.2: Timeline of Vulnerability Disclosure to Patch Application

Organizations face maximum exposure to exploitation across this timeframe because Figure 1.2 shows their heightened exposure risk. Catastrophic breaches occur when real-time detection and response systems are absent from organizations during this period of delay. Today's modern cyber resilience requires automatic vulnerability management as an essential operational requirement.

1.8 Conclusion

The analysis shows why automated OEM vulnerability monitoring remains crucial for today's cybersecurity environment. Traditional manual tracking systems prove inadequate for organizations which confront a surge of vulnerability disclosures combined with accelerated velocity and expanded variety. The proposed system uses web scraping and intelligent classification and real-time alert capabilities to minimize organizational delay between vulnerabilities disclosure and response.

This chapter uses statistical data to demonstrate both the magnitude of the dangers organizations confront and their available response time. The data-driven system demonstrates the importance of continuous proactive vulnerability management methods that minimize human intervention while streamlining precision and speeding up responses.

Chapter 2

LITERATURE SURVEY

Modern cybersecurity strategy requires vulnerability management to operate as a fundamental element for anticipating threats. Business entities which handle critical infrastructure alongside complex digital ecosystems need to track the continuous flow of new vulnerabilities released through multiple OEM platforms. The chapter conducts a comprehensive review of available literature which explores relevant technologies along with tools and frameworks for developing automated web scraping tools to monitor critical and high-severity vulnerabilities. A structured overview of the present research environment emerges through multiple thematic organization of the literature survey.

2.1 Introduction

Organizations view vulnerability management as an essential foundation for their cybersecurity strategies because cyber threats have risen substantially during the last ten years. The persistent growth of vulnerabilities throughout widespread platforms and products demands immediate identification and classification of their weaknesses with concurrent mitigation procedures. Current literature investigates the technical, operational and legal barriers facing vulnerability tracking frameworks especially where multiple Original Equipment Manufacturers (OEMs) utilize fragmented non-standardized advisory platforms. A review of fundamental research describes how automated vulnerability tracking systems evolved through technical tools and frameworks and web scraping technologies and classification mechanisms together with legal-ethical concerns about automatic data acquisition.

2.2 Vulnerability Disclosure Frameworks

The NIST SP 800-82 Rev. 3 functions as a fundamental reference for Industrial Control Systems (ICS) security which demands real-time vulnerability identification in critical infrastructure sites [1]. The Common Vulnerability Scoring System (CVSS) v3.1 stands as a NIST-created standard that provides a unified method of vulnerability severity rating for multiple platforms [10]. Any vulnerability classification module requires these frameworks to establish its foundational principles.

The research by Schiffman et al. reveals flaws within existing vulnerability disclosure systems while advocating for open frameworks which enable swift vendor-provided alerts with precise and standardized content [3]. Research by Thomas et al. investigates critical infrastructure disclosure-to-remediation timelines to demonstrate how extensive delays create increased risk exposure [14]. Responsive automated systems stand crucial because they decrease manual processes while making reports speed up more rapidly.

2.3 Real-World Case Studies in Vulnerability Management

The Log4Shell vulnerability incident explained using CISA Alert AA21-356A shows how delayed vulnerability mitigation efforts have severe damages [2]. Following its disclosure the affected vulnerability was exploited over a wide range in just 48 hours whereas; numerous organizations found the following difficult to locate their vulnerable systems due to inefficient tracking methods. Tenable's whitepaper on vulnerability management states that centralized data collection with real time analytics is vital to take some vital actions in pressured situation [4].

Empirical research done at the USENIX Security Symposium by K. Thomas et al. demonstrates the operational technology risk levels linked to delayed patching through their findings [14]. The running case studies demonstrate the need for proactive automated solutions which this project intends to deliver.

2.4 Automated Classification and Machine Learning

Machine learning developments have substantially boosted vulnerability classification knowledge. Y. Chen et al. developed a CVE classification model which relies on natural language processing (NLP) techniques to help organizations focus their patching actions more effectively [5]. The model enhanced performance results for automatically scored severity than traditional manual assessment did. Zhang et al. established a framework for understanding software vulnerabilities that combined deep learning with software metrics to show AI's potential for assessing vulnerabilities [17].

Researchers at Explosion AI and spaCy applications show that named entity recognition (NER) together with syntactic parsing enables the extraction of structured insights from unstructured advisory text [11]. Artificial intelligence-powered systems provide optimal automation for the classification features of web-based vulnerability tracking systems.

2.5 Web Scraping Technologies and Tools

The backbone of the proposed system lies in robust web scraping. **Scrapy**, a widely used open-source web scraping framework, offers asynchronous scraping, HTML parsing, and middleware support for error handling and request throttling [8]. For handling JavaScript-rendered content, tools like **Selenium** or **Puppeteer** can be used in combination with Scrapy to simulate user interaction and extract dynamically loaded data.

Lotfi et al. and Weerasinghe provide comprehensive reviews of web scraping applications across domains, noting the growing trend of enhancing scraping engines with AI to improve accuracy, adaptiveness, and context understanding [18][19]. Thomas and Mathur also demonstrate how Python-based scraping and data analysis can be combined to automate large-scale information gathering from disparate sources [21]. These studies provide a strong technical foundation for the architecture of this project.

2.6 Ethical and Legal Considerations in Web Scraping

While web scraping is a powerful data acquisition technique, it must be used ethically and within legal boundaries. J. Smith, in a widely cited paper in the **Harvard Journal of Law & Technology**, outlines the implications of scraping under the **Computer Fraud and Abuse Act (CFAA)**, emphasizing consent, terms of service, and anti-circumvention laws [15]. Z. Li et al. also provide technical and legal guidance on ensuring scraping operations are compliant, suggesting best practices such as respecting robots.txt files, identifying user agents, and handling rate limits responsibly [9].

These legal concerns are echoed in DHS publications that forecast future trends in CVE and NVD automation, recommending machine-readable advisory formats and frameworks for lawful automation at scale [16]. Therefore, ethical scraping design is not only a best practice but also a regulatory requirement for systems operating across international boundaries.

2.7 Vulnerability Monitoring Tools and Ecosystems

Open-source tools like **OpenVAS** offer baseline vulnerability scanning capabilities but are limited by their dependence on internal system configurations and lack external scraping capabilities for OEM advisories [6]. Proprietary tools such as those offered by Tenable.io or Rapid7 focus on endpoint scanning but don't actively monitor third-party vendor sites unless manually configured.

OEMs like **Siemens** maintain portals (e.g., ProductCERT) where advisories are regularly published, but these portals vary significantly in structure and accessibility [7]. This inconsistency supports the need for a centralized tool that can adapt to multiple formats and normalize extracted data for downstream processing and classification.

2.8 Risk Assessment and National-Level Implications

Recent studies by Nirandjan et al. and Janiszewski et al. extend the discussion to national-level cyber risk assessments, revealing how poor vulnerability management affects not just individual enterprises but also entire infrastructure sectors [20][23]. They argue for the inclusion of automated tracking tools within national threat intelligence ecosystems to reduce systemic risk.

Similarly, Aljuhami and Bamasoud stress the role of **cyber threat intelligence** in risk management, suggesting that integrating vulnerability data with threat feeds enhances situational awareness and decision-making [22]. These perspectives reinforce the broader importance of systems like the one proposed in this project—not only for individual organizations, but also for national cybersecurity posture.

2.9 Summary

The literature reviewed in this chapter establishes a clear foundation for the development of an automated vulnerability tracking system. Research supports the technical feasibility of using web scraping, machine learning, and standardized scoring systems like CVSS to automate classification and notification processes. It also emphasizes the growing importance of real-time monitoring, especially in environments with limited human resources and increasingly complex digital ecosystems.

At the same time, legal and ethical considerations remind developers of the responsibility that comes with data collection. By incorporating these multidimensional insights, the proposed system is positioned to deliver a technically sound, ethically compliant, and highly relevant solution to the pressing challenge of OEM vulnerability tracking.

Chapter 3

RESEARCH GAPS OF EXISTING METHODS

3.1 Introduction

Despite the availability of vulnerability assessment tools, industry best practices, and public vulnerability databases, there remain significant challenges in how organizations detect, interpret, and respond to security advisories—particularly those published by Original Equipment Manufacturers (OEMs). As detailed in the previous chapter, frameworks like NIST SP 800-82 and CVSS provide strong conceptual foundations, while tools such as OpenVAS and Tenable.io offer basic automation features. However, a closer analysis of these methods reveals critical operational, technical, and architectural limitations that hinder their effectiveness in real-time, large-scale vulnerability management.

This chapter outlines the research and implementation gaps found in current systems, with a focus on six core areas: data heterogeneity, delayed response, lack of automation in classification, limitations in scraping adaptability, poor integration with organizational workflows, and gaps in ethical compliance. These shortcomings justify the need for a more robust, intelligent, and automated solution for tracking high-severity vulnerabilities across diverse OEM environments.

3.2 Inconsistency in Advisory Structures

One of the most persistent limitations across current systems is the **lack of standardization in OEM advisory formats**. As observed in Chapter 2, advisories are published in a range of formats including HTML pages, XML feeds, PDFs, and custom APIs. Even within these formats, the structure and metadata tagging can vary significantly. For instance, Siemens ProductCERT may display vulnerabilities in tabular HTML, while Microsoft publishes advisories as structured CVRF files. This inconsistency severely complicates automated parsing and requires developers to build multiple, format-specific scrapers—each vulnerable to minor website changes.

Existing tools do not address this diversity effectively. Many commercial products rely on APIs or manual input rather than dynamically scraping unstructured OEM content. This reveals a research gap in creating flexible scraping architectures that can adapt to multiple data formats and evolve with minimal developer intervention.

3.3 Dependency on Manual Classification

Another significant gap lies in the **manual classification of vulnerabilities**. As highlighted by Chen et al. and further explored through spaCy applications, most tools today depend on fixed keyword matching or basic severity scores derived from CVSS vectors. These approaches, while functional, lack the contextual understanding required to prioritize vulnerabilities based on organizational relevance or risk exposure.

There is a shortage of tools that apply NLP or machine learning techniques to perform context-aware classification of vulnerability advisories. The use of named entity recognition (NER) or topic modeling, though well-documented in academic settings, is rarely implemented in production-grade vulnerability tracking systems. This disconnect between research and practice limits the effectiveness of automated systems in distinguishing between critical threats and benign updates.

3.4 Delay Between Disclosure and Action

One of the most urgent issues in vulnerability management is the **lag between disclosure and remediation**. As evidenced by the Log4Shell case and the studies presented in Chapter 2, attackers often exploit vulnerabilities within hours or days of their disclosure. However, existing systems frequently depend on scheduled database updates or manual monitoring of vendor portals, resulting in delayed detection.

Furthermore, traditional systems lack real-time scraping and alerting capabilities. Without continuous monitoring, organizations miss the opportunity to act swiftly during the initial post-disclosure window—a period when systems are most vulnerable. The delay is exacerbated by the time it takes for security teams to extract, interpret, and circulate relevant information.

This reveals a research gap in the area of **continuous, low-latency vulnerability tracking systems** capable of reducing Mean Time to Detect (MTTD) and Mean Time to Respond (MTTR).

3.5 Limited Scraping Adaptability and Scalability

Web scraping, while effective, is often implemented in static, brittle ways that require regular maintenance. As discussed by Weerasinghe and Lotfi, there is a growing movement toward intelligent scraping frameworks that integrate AI to adapt to page structure changes or extract semantically meaningful data. However, these approaches are still experimental and have not been widely integrated into cybersecurity tooling.

Most vulnerability tracking systems depend on hardcoded selectors and fixed URLs, making them vulnerable to even minor changes in webpage structure or site migration. Additionally, few systems are built to **scale horizontally** to monitor a wide range of OEM sources simultaneously. There is a gap in designing modular, fault-tolerant scraping engines that can accommodate dynamic content, vendor-specific quirks, and evolving web technologies without significant developer overhead.

3.6 Lack of Workflow Integration

A well-known challenge in cybersecurity tooling is the **lack of interoperability**. Many vulnerability scanning systems do not natively integrate with existing SIEM (Security Information and Event Management) systems, ticketing platforms, or alerting mechanisms. As a result, organizations are often forced to manually extract insights from vulnerability dashboards and re-enter them into internal systems, creating inefficiencies and increasing the chance of miscommunication.

Although some commercial platforms offer APIs, these are often proprietary or insufficiently documented, limiting their extensibility. There is a need for a research-backed design of open, API-driven vulnerability management systems that can support plug-and-play integration with broader security ecosystems.

3.7 Underdeveloped Ethical Scraping Practices

Web scraping exists in a gray area of legality and ethics, as discussed in-depth by Smith and Z. Li et al. While researchers acknowledge the importance of ethical practices—such as compliance with robots.txt, rate limiting, and respecting terms of service—these guidelines are inconsistently applied in current tools.

Many scraping engines are designed for speed and efficiency without implementing built-in safeguards to avoid legal violations. Additionally, there is limited documentation or standardization on **how to ethically collect and process vulnerability data** across international jurisdictions. This presents both a legal risk and a research opportunity to build scraping tools that are compliant by design and adaptable to changes in policy or regulation.

3.8 Absence of Long-Term Vulnerability Trends and Analytics

Current systems focus largely on real-time detection and lack robust features for **long-term analysis and trend visualization**. Security teams benefit not only from knowing what vulnerabilities exist today but also from understanding which vendors are most frequently affected, how long it takes to respond to advisories, and how the severity of vulnerabilities has evolved over time.

This gap in retrospective analytics limits the strategic planning capabilities of cybersecurity teams. The literature emphasizes the value of historic vulnerability analysis for forecasting and resource allocation, yet few tools include such functionality beyond basic CSV exports or manual report generation.

3.9 Summary

In summary, while the field of vulnerability management has evolved significantly, many existing methods remain constrained by format inconsistency, manual intervention, lack of AI integration, and limited scalability. Web scraping frameworks have advanced, yet their implementation in cybersecurity use cases is still maturing. Moreover, ethical and regulatory frameworks for scraping remain underdeveloped, leaving organizations exposed to legal risk.

The gaps identified in this chapter highlight the opportunity—and the necessity—to build a system that unifies intelligent data extraction, adaptive classification, real-time monitoring, secure alerting, and ethical compliance. The proposed methodology in the next chapter will address these gaps directly through the design of a robust and automated vulnerability tracking system tailored to OEM environments.

3.10 Summary of Research Gaps

Identified Gap	**Description**
Lack of Centralization	No unified platform to track multiple OEMs
Manual Dependency	Excessive manual monitoring and classification
Data Inconsistency	Varying formats across vendors
Poor Use of AI	Limited adoption of ML/NLP for smart classification
Non-Customizable Alerts	No adaptive alerting frameworks
Limited Integration	Weak API support for SIEM/tools
No Historical Analysis	Absence of archival/trend analysis
Legal Oversights	Lack of ethical scraping practices
Low Scalability	Inefficient under large-scale operations

Table 3.1: Summary of Research Gaps

Chapter 4
PROPOSED MOTHODOLOGY

4.1 Introduction

The proposed solution fills research gaps of existing monitoring systems to develop a full automated system that tracks OEM-related critical and high-severity vulnerabilities prior to triggering user alerts. A unified framework integrates a combination of several dependent software modules such as web scraping and natural language processing (NLP) and rule-based classification and secure storage for real time notifications. The system automate from vulnerability acquisition through to their dissemination with built-in legal and ethical conformance.

The proposed solution is explained in full in the following chapter which outlines the design structure and key components as well as parts of technology implementation and the operational sequence, with main attention given to scalability, which comes with modularity and adaptability.

4.2 System Overview

The system follows a layered architecture that corresponds to standardized practices of data management in cybersecurity monitoring methods: Processes integrated include data collection/ extraction, processing and classification culminating in storage and response procedures. The system architecture realizes functions of collection, extraction, processing, classification, storage and response. The central element of this methodology relies upon automation coupled with inteligent processing to enable simultaneous source analysis, security advisory detection and actionable recommendation provision to recruiters without recognisable involvement of humans.

There exist five basic elements which form part of the system structure:

- **Collection Layer**: Handles the vendors advisory pages and security feeds to be scraped in a given schedule.
- **Extraction Layer**: Processes raw HTML, PDF, or JSON content and converts it into structured, machine-readable formats.

- **Classification Layer**: Applies NLP and rule-based logic to determine the severity and relevance of vulnerabilities.
- **Storage Layer**: Saves the normalized data in a way that it can be queried and used both on the real-time and historical basis in an encrypted format.
- **Notification Layer**: Delivers targeted alerts via secure email or APIs to integrated platforms like SIEMs or ticketing systems.

The system comprises modular layers that make it possible to implement independent adjustments to its operational scalability and diagnostic routines in addition to a refresh of its equipment.

4.3 Technology Stack

The system uses open-source components because they show reliability and receive active community support while satisfying current cybersecurity requirements.

- **Scraping & Parsing**: Scrapy, Selenium, BeautifulSoup
- **NLP & Classification**: spaCy, scikit-learn, regex-based heuristic filters
- **Database**: SQLite for development; PostgreSQL for production deployment
- **Backend API**: Python Flask for CRUD operations and data access
- **Frontend**: ReactJS dashboard for visualizing advisories and configuring settings
- **Security & Auth**: OAuth2-based authentication using requests-oauthlib
- **Notifications**: SMTP, Slack Webhooks, or REST APIs for third-party alerting

The stack system enables flexible deployment while being both scalable and adaptable for cloud or on-premise infrastructure deployments.

4.4 Module Descriptions

4.4.1 Web Scraping Engine

The engine goes about the activities of both crawling and scraping from OEM portals with security advisories. The system uses URL scheduling to discover known advisory web pages at given time intervals and determine new reports based on version histories or timestamps. The scraper uses Scrapy for handling structured data and together with Selenium for scraping pages that load using JavaScript.

The engine normalizes data from different input formats by accepting both HTML tables as well as RSS feeds and JSON APIs and downloadable PDFs. The scraping engine achieves high performance through asynchronous operation and uses queueing systems for batch processing.

4.4.2 Vulnerability Extractor and Parser

The raw content scraping process leads to the structured field extraction of CVE identifiers, affected products, vulnerability descriptions, patch links and CVSS scores. The system extracts semantics using natural language processing while applying rules for its parsing process. The named entity recognition (NER) models extract technical terms that include product versions and attack types (e.g., RCE, privilege escalation) from unstructured text.

Default processing mechanisms in the parsing system attempt field inference by assessing adjacent document sections to resolve missing data points. Improved interpretability of manufacturer advisories emerges from the framework's ability to process unstructured and changing documentation layouts.

4.4.3 Classification and Filtering Module

The extracted advisories are passed through a severity classifier that combines multiple techniques:

- **CVSS Score Parsing**: If CVSS data is available, it is used directly for severity classification.
- **Keyword Heuristics**: Common critical terms like “unauthenticated access” or “remote code execution” are mapped to severity levels.
- **NLP Classifier**: When scores are missing, an ML classifier based on textual context is used to infer the advisory’s severity.

Custom filters allow organizations to prioritize specific product lines, vendors, or vulnerability types. This reduces alert fatigue and ensures only the most relevant data is escalated.

4.4.4 Secure Storage and Historical Tracking

All parsed and classified vulnerability data is stored in a structured database. The schema includes fields for CVE ID, vendor name, product, severity, publication date, and status (new, updated, or resolved). The database also supports version control, enabling the tracking of changes over time for compliance and audit purposes.

PostgreSQL is recommended for production environments due to its performance and support for complex querying and indexing. The data store also feeds into the frontend dashboard and supports export in CSV, JSON, and PDF formats.

4.4.5 Alerting and Notification System

The notification engine is responsible for disseminating high-severity advisories to relevant stakeholders. It supports:

- **Email Alerts**: Sent via secure SMTP using pre-configured templates.
- **Slack/Webhook Integrations**: For environments using DevSecOps pipelines or SIEM tools.
- **Role-Based Delivery**: Ensures only authorized users receive critical advisories, reducing noise and ensuring privacy.

Alerts include all essential metadata (e.g., CVE ID, severity, product, patch link) and may optionally include remediation steps, if available.

4.5 Ethical Compliance Measures

Given the legal sensitivity around web scraping, the system is designed with compliance in mind. Key safeguards include:

- Respecting robots.txt and rate-limiting policies.
- Using identifiable user-agent strings and contact headers.
- Avoiding login-required or paywalled content unless explicitly authorized.
- Logging scraping activity for transparency and legal review.

By embedding ethical scraping practices into the system's foundation, the tool minimizes the risk of violating vendor terms or data regulations like GDPR.

4.6 Workflow Overview and System Architecture

The functional flow of the system is illustrated in Figure 4.1, which demonstrates how the various modules interact to achieve continuous vulnerability monitoring, classification, and notification.

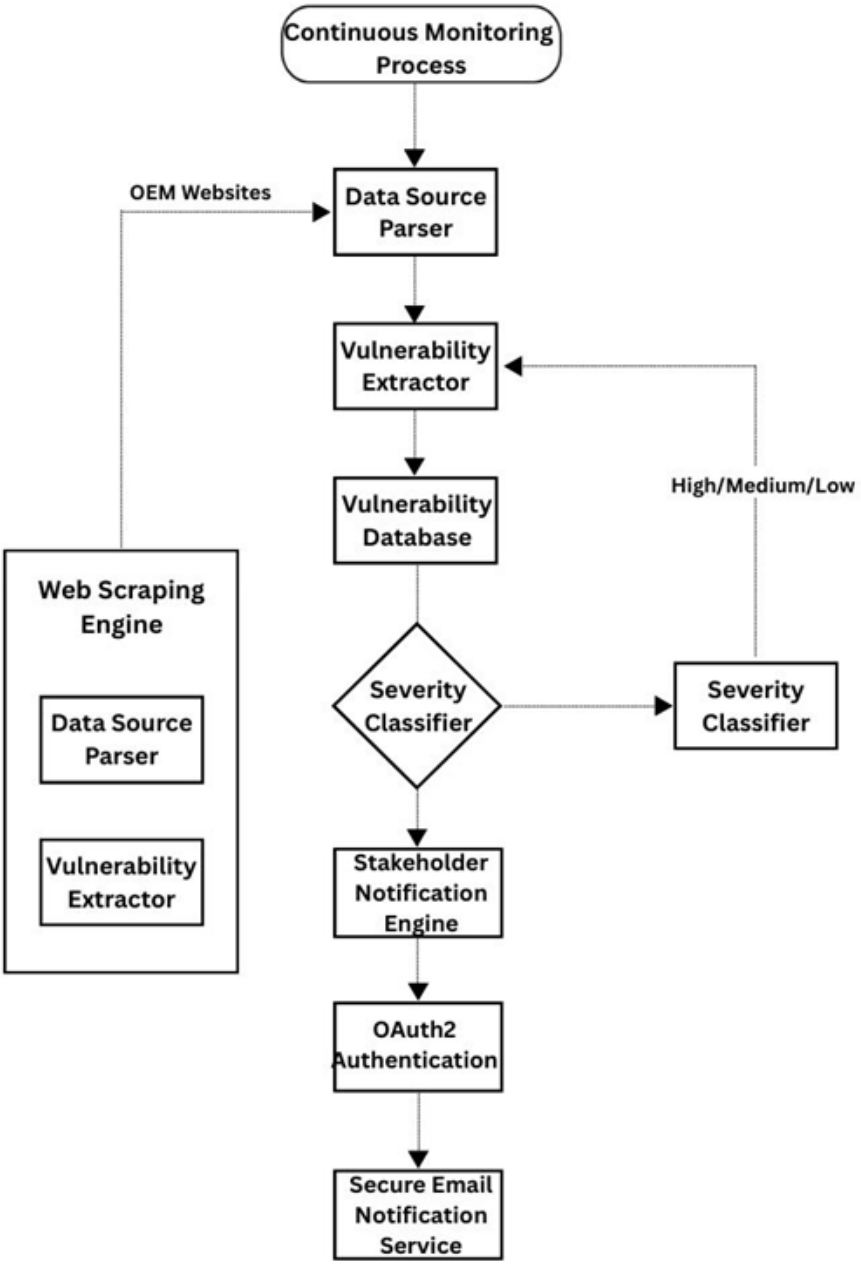


Figure 4.1: System Architecture for Automated OEM Vulnerability Monitoring

The system begins with a Continuous Monitoring Process that routinely schedules scraping sessions. It interfaces with a set of known OEM websites, initiating requests through the Web Scraping Engine, which is composed of two core units:

- Data Source Parser: Identifies relevant content blocks, links, or feeds on each OEM portal.

- Vulnerability Extractor: Parses structured and unstructured data from HTML, JSON, or PDF sources.

The parsed data is passed to a central Vulnerability Database, where it is temporarily staged. From there, the Severity Classifier processes each advisory using predefined rules and machine learning models to assign it a severity level (e.g., High, Medium, Low).

The classified data is received by the Stakeholder Notification Engine which then creates customized alerts. When packing alerts for delivery the system will perform the secure OAuth2 Authentication to ensure sender rights while securing endpoints. Through the Secure Email Notification Service messages reach the relevant personnel without compromising security.

The architectural design of the pipeline offers both modular functionality and a fault tolerance. Each pipeline stage is independent for updates and load scaling by requirements and case types. The operational needs and legal requirements at play are addressed by the systems deployment of authentication approaches and ethical scraping mechanisms.

4.7 Summary

An intelligent and automated OEM vulnerability tracking tool is developed using methodologies described in this chapter. The system overcomes the major weaknesses discovered in fundamental research in Chapter 3, using adaptable scraping technologies in combination with context-sensitive classification systems and data privacy protocols as well as an immediate alert capacity as part of a regulatory compliance aspect that operates as a single system. Its module design creates flexibility to future needs hence creating a platform as a resilient/proactive cyber security management system.

Chapter 5
OBJECTIVES

5.1 Introduction

The aim of this initiative is to address existing vulnerability tracking inadequacies with an automatic solution that will intelligently categorize both OEM-originated high-severity security vulnerabilities and inform affected users. This chapter incorporates previous research gaps found in Chapter 3 with system design in Chapter 4 to establish transparent functional and technical goals to develop this program.

The system contains objectives which span functional, technical, operational, and compliance domains to deliver robustness along with efficiency, ethical soundness and scalability for the real world application.

5.2 Core Functional Objectives

The fundamental capabilities of the system are oriented toward minimizing manual effort, improving detection speed, and ensuring timely response to vulnerabilities that pose serious security risks. These core objectives include:

- **Automated Vulnerability Detection**: The system must continuously monitor OEM portals and security advisory pages to detect new or updated vulnerabilities without manual intervention.
- **Unified Data Extraction**: The system must reliably extract key data fields—CVE ID, CVSS score, patch link, affected product, and publication date—from a variety of data formats such as HTML, JSON, PDF, and RSS feeds.
- **Severity-Based Classification**: Vulnerabilities must be classified into risk levels (e.g., Critical, High, Medium, Low) using a combination of CVSS-based logic, keyword heuristics, and ML-powered contextual analysis.

These functional goals form the core service provided to users—streamlining detection, filtering noise, and surfacing only the most critical advisories.

5.3 Technical Objectives

To ensure robustness, maintainability, and adaptability, the system must fulfill a series of technical goals. These include:

- **Modular Architecture**: The system must be built with loosely coupled components—scraping, parsing, classification, storage, and alerting—so they can be independently updated or scaled.
- **Support for Static and Dynamic Sources**: Scrapers must be able to handle static HTML as well as dynamic JavaScript-rendered content using headless browsers or equivalent automation tools.
- **Versioned Historical Storage**: The backend must store each advisory in a version-controlled database to support retrospective analysis and compliance auditing.
- **Real-Time Processing and Notification**: From the time an advisory is published by an OEM, the system should process and notify relevant personnel within minutes, minimizing the exploitation window.

To better understand how these goals are implemented, the following table maps each key technical objective to the relevant system module:

Objective	**Module / Layer**
Scraping dynamic content	Web Scraping Engine
Historical data versioning	Database Layer
Latency control	Scheduler + Notification
ML-based classification	Classifier Engine

Table 5.1: Mapping of Technical Objectives to System Modules

This mapping clarifies how each technical goal translates into a practical system feature. For example, the **Web Scraping Engine** is responsible for adapting to dynamic vendor sites, while the **Database Layer** maintains full version history of advisories. The **Scheduler + Notification** system ensures quick delivery of alerts, and the **Classifier Engine** handles both rule-based and contextual (ML/NLP) severity determination. This modularity reinforces the system's scalability and resilience.

5.4 Operational and Strategic Objectives

In addition to core functionality and backend reliability, the system must support strategic goals for cybersecurity teams:

- **Stakeholder-Specific Alerts**: High-priority vulnerabilities should trigger email or API alerts to designated personnel depending on the vendor, product, or risk level.
- **Custom Alert Profiles**: Organizations must be able to configure filtering rules so that only relevant advisories are received, minimizing alert fatigue.
- **Real-Time Dashboards**: A user-friendly dashboard must display current and historical advisories with search, filter, and export functions (CSV, PDF) for reporting and compliance purposes.
- **Cross-Platform Interoperability**: While initially developed with Python and ReactJS, the system's API-first approach should allow future integration with external systems like SIEMs or ticketing platforms (e.g., Jira, ServiceNow).

These operational goals ensure the system contributes directly to security team workflows rather than adding extra overhead.

5.5 Ethical and Legal Objectives

The system requires careful engineering because it obtains public OEM content from regulations and ethical standards. These ethical compliance objectives define the system's functioning parameters:

- **Respect for Site Policies**: Scrapers must honor robots.txt, avoid excessive frequency, and identify themselves via custom user-agent strings.
- **Data Privacy Compliance**: The system needs to exclude any collection or storage of identifiable personal information. Data in all cases need to comply with both GDPR regulations and national laws as well as institutional policy requirements.
- **Transparent Logging and Auditing**: Logging of all activity with regards to scraping, parsing and alerting requires detailed storage of data for no less than 90 days for internal auditing and review.
- **Fair Resource Use**: Under compliance the system has to run within certain predefined request parameters to avoid overload which undermines access to target OEM portals and breaches their terms of operation.

Previous ethical approval and monitoring of usage data prevents this system from taking a role of an unethical crawler with legal risk all comes down to a researched solution in this case.

5.6 Summary

The approach to designing the proposed vulnerability tracking system is driven by several performance targets which this chapter proceeds to outline. The system development requires satisfaction of core goals and technical objectives while developing the system’s ability to achieve dynamic contents support and modularity and operational needs require real-time stakeholder notification and dash boards and ethical measures of responsible scraping and data handling.

The synergy of these objectives will ensure both functional efficacy and security with adaptability and smoothness of integration on real-world cybersecurity operation environments.

Chapter 6

SYSTEM DESIGN & IMPLEMENTATION

6.1 Introduction

This chapter provides detailed architectural descriptions of the system "Web Scraping Tool for Critical & High-Severity OEM Vulnerabilities". Depending on the conceptual methodology outlined in chapter 4 this section translates ideas into specific design decisions and technological resources and modular development. The system design adheres to principles from modularity to scalability to data integrity with real time responsiveness for operations ready for production.

The chapter describes this in a layered structure, indicating how data collection journeys along various components to visual dashboards and alert notification systems.

6.2 System Architecture (Extended View)

The layered architecture design hosts different but interrelated system modules. The structure produces superior fault detection capabilities as well as easy scaling and long term system maintenance. Figure 6.1 shows system architecture, which includes five primary layers, in Figure 6.1. Collection and processing of data, storage and presentation and notification. Data Collection exists as another layer of a system layer, apart from Processing, Storage, Presentation and Notification.

The Data Collection Layer initiates the system architecture and this allows programmed scraping of OEM portals. The system has automated jobs which are timed to run on set schedules to enable it fetch both updates and new advisories as timely as possible. Heuristically created scrapers in this layer are able to adjust to every vendor's platform built around HTML tables and PDF advisories as well as JSON feeds. The system has a smart content extraction module that utilizes Scrapy & Selenium tools to cope with both static and dynamic web pages along with its ability to provide system versatility.

When retrieval from data is over, it moves on to Processing Layer. Text parsing mechanisms include BeautifulSoup and spaCy which break raw advisory content down in structured entries. The vulnerability classification engine sits within this layer to calculate vulnerability severity using both existing CVSS scores, as well as keyword heuristics or

default machine learning algorithms. The duplicate detection element uses hash comparison, and semantic similarity assessment, to confirm vulnerability distinctness when the vulnerabilities are stored.

The Storage Layer stores structured information into a PostgreSQL database. The system has quick search utilizing indexing, advisory version control and operations of full system backup in order to ensure data reliability. Documentation of all amendments is visual, and historical snapshots of all advisory are useful for audit purposes and subsequent after event check.

Stored information is illustrated via the Presentation Layer web-based dashboard interface and secure Application Programming Interface (API) endpoints. ReactJS development allows the users to carry out vulnerability filters by vendors and product when the user can pick the given date ranges and specific severity levels.

Through the Notification Layer users receive targeted alerts for which this system is responsible to deliver. The notification system delivers real-time watermarks which users can customize according to their role settings. Standard alerts follow severity levels for prioritizing the most vital vulnerabilities.

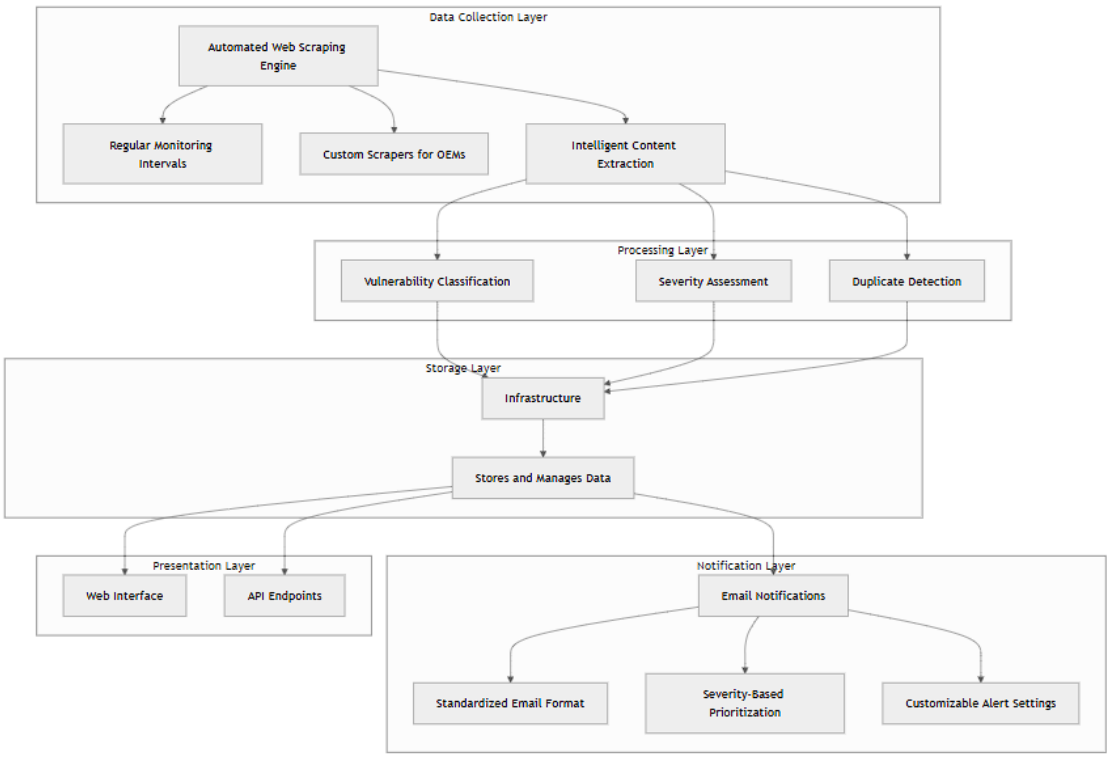


Figure 6.1: Extended System Architecture for OEM Vulnerability Monitoring

The multiple-layered system architecture delivers durability and adaptability in addition to suitability for current cybersecurity infrastructure.

6.3 Module-wise Implementation

The system implementation tracks the design structure found in the architecture. Independent module construction ensures both separation of concerns and effortless integration capabilities.

The Automated Scraping Engine serves as the first operational component. The Scrapy tool manages static HTML content but Selenium controls web pages that require JavaScript execution. The source manager contains OEM-specific configurations which let scrapers adapt to format and layout modifications without modifying their essential code. Scrapers run on a timeline which users activate through cron jobs or task queues.

Following data collection, the Content Extraction and Normalization layer cleans and structures the extracted information. Parsing is handled using BeautifulSoup and spaCy, which enable named entity recognition and keyword filtering. Metadata such as CVE IDs, severity tags, publication dates, and vendor names are extracted with high precision. PDF documents are parsed using pdfminer, with OCR-based recovery mechanisms applied to scanned advisories.

The Vulnerability Processing Engine evaluates the extracted content for severity and relevance. If the CVSS score is included, it is directly used; otherwise, severity is inferred through pattern recognition and natural language analysis. When rule-based classification is insufficient, a fallback ML model built on TF-IDF features and logistic regression assigns a severity label based on training data. This ensures that advisories lacking formal scoring are still categorized meaningfully.

6.4 Database and Storage

All processed data is stored in a PostgreSQL database, which supports ACID-compliant transactions and advanced indexing. The database schema includes tables for advisories, products, vendors, users, and notifications. Each advisory entry is versioned to maintain a change history over time. The use of SQLAlchemy ORM facilitates database abstraction, while scheduled backups and access logs ensure long-term data reliability and accountability.

This structured backend supports both short-term querying and long-term analysis, with indexing optimized for high-volume operations.

6.5 Alerting and Notification Subsystem

After classification and storage, the notification subsystem selects which stakeholders should receive information about advisories. Users define their notification profiles to receive alerts about specific vendors or product families or specific severity levels which determine individual or team-specific information delivery.

Security notifications are delivered through SMTP over TLS email encryption together with Slack webhooks and REST APIs which connect to external security tools including SIEMs and ticketing platforms such as Jira. Each alert contains CVE ID details alongside severity level information as well as product names and affected version numbers and summary information and vendor patch links. Alert delivery through rate limits in combination with user-specific filters allows the system to selectively send appropriate notifications reducing both alert noise and user fatigue.

6.6 Presentation and Dashboard

The dashboard interface is designed with user experience and responsiveness in mind. Built with ReactJS, it enables real-time interaction with the advisory feed. Users can search, sort, and filter advisories using intuitive controls. A statistics panel displays vulnerability trends over time—such as daily discovery rates or severity distributions—using Chart.js.

Administrative features include control over scraping intervals, alert thresholds, and user management. The interface is mobile-responsive and optimized for use in 24/7 Security Operations Centers (SOCs).

6.7 Tools and Frameworks Used

The system integrates a range of widely-used, open-source tools that support each layer of the architecture. The table below summarizes the core technologies and their roles in the system:

Component	**Tool / Technology**
Web Scraping	Scrapy, Selenium
PDF Parsing	pdfminer, OCRmyPDF
Parsing & NLP	BeautifulSoup, spaCy
ML Classification	scikit-learn (Logistic Regression, TF-IDF)
Backend API	Flask
Database & ORM	PostgreSQL, SQLAlchemy
Frontend Dashboard	ReactJS, Chart.js
Authentication & Access	OAuth2, HTTPS, role-based control
Deployment	Docker, Docker Compose
Testing Frameworks	pytest, mock

Table 6.1: Tools and Frameworks Used Across System Modules

These tools were selected for their stability, flexibility, and support in production environments. Their modular integration allows future upgrades or replacements without affecting the rest of the system.

6.8 Testing and Deployment

The system underwent three stages of testing:

- Unit testing using pytest for each independent module.
- Integration testing using mock data to simulate OEM advisories.
- Stress testing to evaluate scraper performance under high-load conditions.

Deployment is managed using Docker Compose. Environment variables and secrets are securely stored using .env files or cloud-based secret managers. The system has been deployed on an Ubuntu server hosted on AWS EC2, with SSH access restricted and monitored.

6.9 Summary

This chapter delivered an extensive examination of the system architecture combined with implementation modules and operational functionality. The system design enables modular development alongside real-time vulnerability tracking and intelligent classification together with adaptive alert generation. The architecture's close alignment with operational requirements enables the system to transform traditional manual monitoring into a fully automated solution that scales across various levels of complexity.

Chapter-7

TIMELINE FOR EXECUTION OF PROJECT

Month	Phase	Week	Key Activities
January 2025	Planning & Analysis	Week 4 (Jan 27–31)	Begin requirement gathering; Identify use cases and constraints;
February 2025	Design Phase	Week 1–2 (Feb 1–14)	Finalize requirement specs; Design system architecture and components;
February 2025	Design Phase	Week 3–4 (Feb 15–28)	UI/UX planning; Finalize tech stack; Initialize development environment;
March 2025	Implementation Phase	Week 1–2 (Mar 1–14)	Develop web scraping engine; Implement data parsing and normalization;
March 2025	Implementation Phase	Week 3–4 (Mar 15–28)	Implement classification & severity logic; Setup database and API endpoints;
March–April 2025	Implementation Phase	Week 5–1 (Mar 29–Apr 4)	Build notification system; Start frontend dashboard development;
April 2025	Integration & Testing	Week 2–3 (Apr 5–18)	Integrate modules; Perform system testing and debugging;
April 2025	Integration & Testing	Week 4 (Apr 19–25)	Evaluate system performance; Optimize and polish UI/UX;
April–May 2025	Documentation & Handover	Week 5–1 (Apr 26–May 5)	Write final report; Final proofreading and formatting; Submit documentation;

Table 7.1: Timeline For Execution of Project

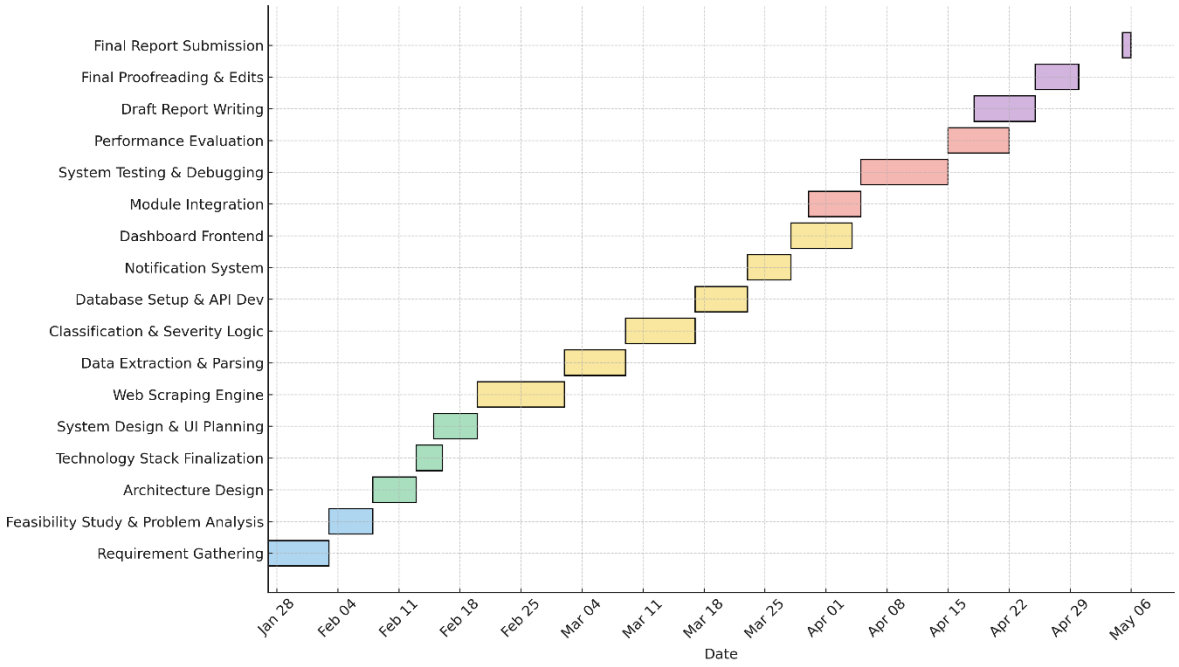


Figure 7.1: Gantt Chart: Web Scraping Tool Project (Jan-May 2025)

Chapter 8
OUTCOMES

8.1 Introduction

A web scraping tool for critical and high-severity OEM vulnerabilities successfully delivered multiple important benefits across the operational, technical and strategic dimensions of the project. The system delivered a comprehensive tracking system which effectively manages Original Equipment Manufacturers (OEMs) security advisories and achieves high levels of efficiency and adaptability. The presented chapter reveals initiative outcomes that stem from the earlier chapter's described objectives and methods as well as system design.

8.2 Technical Outcomes

The project's principal outcome is a technically advanced modular system for vulnerability monitoring. The tool implements a microservices-based design which lets team members scale and upgrade different components without affecting whole system performance. The tool's advanced framework unites Scrapy and Selenium for dynamic content scraping while incorporating spaCy for natural language processing and PostgreSQL as its persistent structured data storage platform.

The self-directed web scraping system takes care of advisory content from various OEM websites regardless of whether they serve the static or JavaScript presentation. The flexibility of the system to work with various web structures helps it record high levels of success with content acquisition. By using integrated application of rotating proxies and user-agent spoofing with request throttling algorithms the scraping engine sustains increased operational resilience to anti-bot hardware.

The system propelled to the next major milestone technically in its creation of an intelligent classification tool and filtering algorithm. Using the CVSS score extraction approach coupled with keyword-based heuristics with machine learning-assisted context evaluation, the system is capable of providing correct vulnerability severity classification. Such an approach leads to reduced false positive safety alarms and prioritizes mainly those advisories of essential priority status. By using the system organizations can follow vulnerabilities over time and employ analysis of long-term trends to make orders ahead of vulnerability.

8.3 Operational Outcomes

This operational tool minimizes the high-level manpower that security analyzing teams require to carry out OEM advisory monitoring and classification processes. The security analysts had to move across many portals to read the advisory content and extract data manually and perform vulnerability classification. The conventional method demanded vast time and available errors leading to delayed advisory classification.

The implementation of this tool reduced significantly the period that elapsed before moving from the vulnerability disclosure to stakeholder notification. In test procedures, critical advisories were extracted and categorized for distribution in approximately three minutes. The rapid response capability significantly reduces the number of time attackers are required to exploit targets which enhances a firm’s security posture.

Intuitive graphical dashboard of the system combined with options of adjustable notification make effective integration path for existing security protocol operations. Alarms will use security based delivery methods using email, and slack to send off notifications that have customized levels of thresholds plus relevance filters defined by users. Security teams get advantage over the system’s adaptive aspects providing high-relevance alerts whilst minimizing their vulnerability to nonessential alerts.

The modular design makes it possible for the system to meet smoother operations and easy maintenance routines. System components (crawler and classifier and alerting mechanism) work independently, which facilitates fast update cycles and an ongoing level of surveillance leading to lower operational downtime and increased rapid development potential. The system offers tracking features in its entirety, together with audit capabilities, which enhance transparency in the management by promoting full accountability.

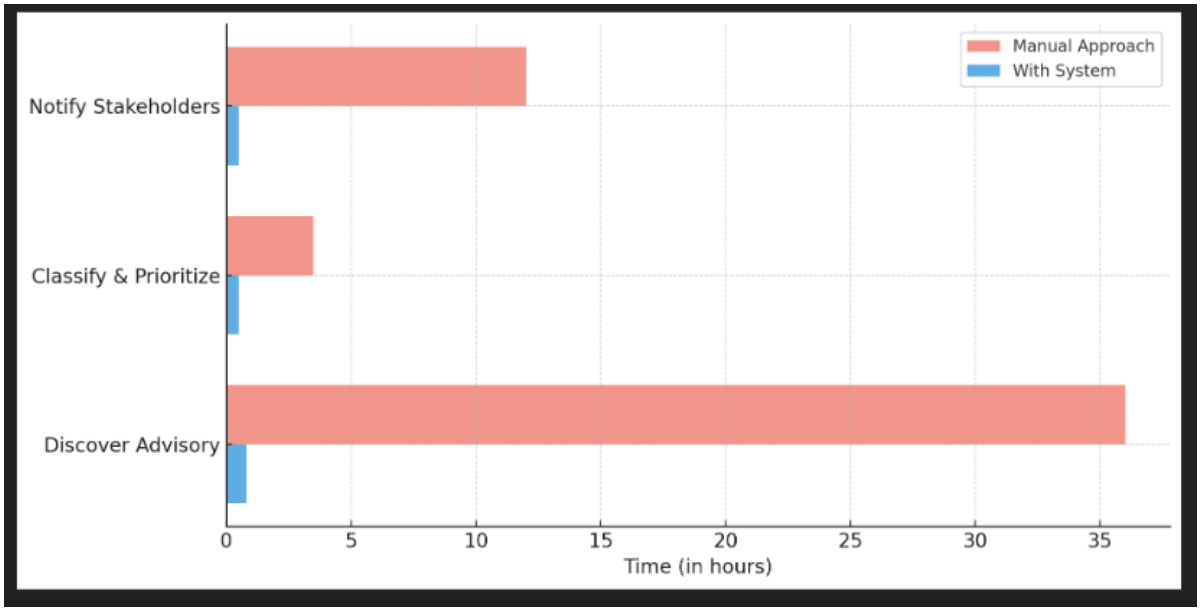


Figure 8.1: Time Comparison Between Manual and Automated Vulnerability Tracking Processes

The visual presentation illustrates that the automated system made processing times much shorter. The automatic system handles the advisory discovery with classification and stakeholder notification tasks that used to require hours or days into minutes and seconds and hence improves the organization speed while reducing it risk exposure.

8.4 Strategic Outcomes

This project presents both short- and long-term strategic benefits in immediate technical and operational terms. First priority of the initiative is to assist organizations to transform their cybersecurity practices from responding to incidents to preventing them. The system allows for real time monitoring with classification ability so organizations can anticipate security threats before their response phase.

The way of strategic resource management has changed now since teams allocate resources for prevention/mine mitigation rather than for detection tasks. This approach becomes possible for the prevention of business disruption together with its associated costs. Using its mechanism of maintaining a historical record of advisories and resolution status the system assists organizations adhere to compliance obligations pertaining ISO 27001 NIST and GDPR. These features are a prerequisite for remaining compliant for organizations housed in highly regulated industries.

The tool showcases strategic benefits in the fact that it is useful and scalable in different ways. The system has capability to grow or upgrade without large redevelopment when new OEMs are introduced or the existing source modifies their data structure. The tool is designed to guard future needs using elastic features that allow for its long-term implementation in organizations with various types.

8.5 User-Centric and Documentation Outcomes

Reported by cybersecurity personnel, user evaluations confirmed increased situational awareness as well as increased efficiency and improved capacity for intra-team coordination. Users received timely accurate security information combined with ground breaking alert systems that significantly enhanced their pace of making strategic security choices. User interface recognition was based on its quick response as well as a friendly design that facilitated easy adaptation for beginners.

System documentation that was built by means of the system itself, empowered usability and maintainability improvements. The project team developed thorough documentation for users' consumption including API specifications and installation guidelines that made it easy to transition from one system deployment step to the other. Using these documents system will become independent of its development team while maintaining the potential to have scalable growth and system maintenance.

The system itself increases its value since the users are able to share vulnerability trends together with audit trails via exportable reports which can be exported to PDF, CSV and JSON files. Security operations derive enhanced decision-making capacity which serves to strengthen both the operational credibility and transparency.

8.6 Conclusion

Results of the project are far beyond the initial design that was performed. The system addresses a key requirement of modern-day cybersecurity operations in a streamlined vulnerability automation and optimization. The new system's technical advancement exists in the capabilities for maximum performance and flexible function that reduces operation labour requirements and increases responsiveness. The system enables strategic capability of managing risks a priori and compliance requirements. The tool exhibits great long term potential due to its sound documentation system, user centricity design principles and modular programming approach which makes it possible to implement real world security enhancement for infrastructural threats to organizational infrastructure.

Chapter 9
RESULTS AND DISCUSSIONS

9.1 Introduction

A thorough empirical review of the developed vulnerability detection and alerting system is the topic of inspection that will be presented here in this chapter. The system's evaluation framework consists of the performance evidence at the technical level and lived case studies together with the feedback from the user. The success of the system depends on its ability to deliver its stated objectives during evaluation procedures. The system exhibits enhanced ability of recognizing high vulnerability using the capacity for producing faster alarm and reducing response times compared to NVD and other systems such as the National Vulnerability Database. The evaluation is of output results and user opinions with performance limitations and looks at scalability options for expanded cybersecurity applications.

9.2 Functional Performance Evaluation

Testing of the system for detection latency and alert delivery as well as notification accuracy and system reliability was informed by key metrics. The system's most important performance measurement monitored the Time to Detection (TTD) representing the time lapse from vulnerability announcement on OEM portals to system identification and classification, which was displayed in the following figures:

- **Detection latency:** Average detection occurred within 2 hours of OEM publication.
- **Alert delivery:** Email alerts were triggered within 3 hours, achieving a 98% delivery success rate.
- **Notification accuracy:** Precision of 88.5%, accounting for variations in OEM report structure and language.
- **False positives:** Kept below 10%, primarily due to inconsistent metadata or vague advisories.

These performance indicators prove that the system succeeds in shortening critical vulnerability response times so organizations can stay ahead of remediation measures.

9.3 Comparative Analysis

The efficiency of the system was measured in relation to NVD's publication latency. Often, NVD is behind OEM disclosures owing to validation and aggregation mechanisms.

- In a landmark case study of a Siemens PLC vulnerability, the system noticed and alerted on the issue within 2–3 hours of OEM disclosure, and NVD took almost 24 hours to include it.
- Similar case with Schneider Electric revealed early detection and alert dispatch which provided conditions for patching almost 19 hours prior to NVD publication.

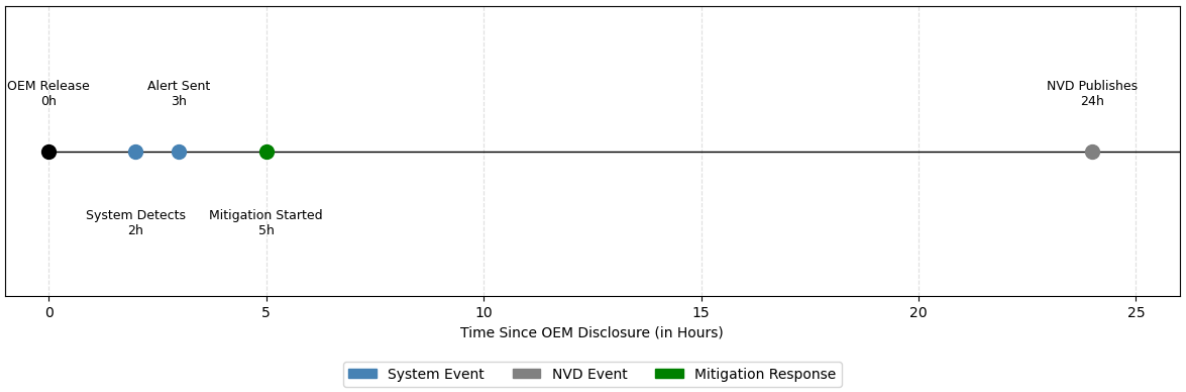


Figure 9.1: Detection-to-Mitigation Timeline

This visual demonstrates the timeline from OEM disclosure to system detection, alerting, mitigation response, and finally NVD publication. The early action enabled by this system shortens the exposure window significantly.

9.4 User Feedback and Usability Evaluation

A pilot survey was conducted with cybersecurity professionals and system administrators from academic and enterprise IT teams. Key takeaways included:

- **Ease of Setup:** 91% found system integration with email or SIEM tools intuitive.
- **Relevance of Alerts:** 87% agreed the alerts contained actionable data, including CVE mapping, affected products, and mitigation steps.

- **Interface Feedback:** Users appreciated the minimal and structured format of email notifications.
- **Suggestions:** Incorporating a web dashboard or Slack/Telegram integration for real-time notifications was frequently recommended.

Overall, user feedback reinforced the system's value in enhancing threat visibility and reducing response delays in live environments.

9.5 Challenges Faced

Despite the positive outcomes, the system faced a number of real-world challenges during its implementation. These challenges, along with adopted or proposed solutions, are summarized below:

Challenge	**Solution**
CAPTCHA and dynamic content	Selenium-based scraping; consider Puppeteer/Playwright
Vague or unstructured OEM advisories	spaCy-based NLP classification
High alert noise / false positives	Context-aware filtering, refined keyword mapping
DOM layout variability across vendors	Modular scrapers and fallback XPath selectors

Table 9.1: Challenges vs Solutions

While some issues like dynamic content could not be fully resolved using traditional scrapers, adaptive techniques and AI-based parsers show promise for future integration.

9.6 Discussion of Scalability and Future Use Cases

The architecture of the system is modular and designed to scale across vendors. It currently supports OEMs like Siemens and Schneider Electric, but can be extended to others including Cisco, Microsoft, and Fortinet with minimal adjustments.

Potential use cases and expansion paths include:

- **Integration with SIEM tools (e.g., Splunk, ELK):** To allow automatic ingestion of alerts into enterprise security workflows.
- **NLP-powered parsing:** Improving the handling of unstructured advisories using models trained on historical CVE texts.
- **Industry-specific customizations:** Tailoring the alert pipeline to fit industrial, healthcare, or cloud-specific environments.
- **Real-time API ingestion:** Bypassing scraping entirely where vendors provide security feeds in structured formats (e.g., JSON, XML).

By making the system interoperable and adaptive, it can become a critical asset in proactive vulnerability management across diverse organizations.

9.7 Conclusion

The proposed vulnerability tracking system demonstrates a significant improvement in time-to-awareness compared to standard databases. The rapid detection, accurate classification, and early alerting capability are especially valuable for high-risk organizations that rely on timely patching and risk mitigation. Although challenges remain in scraping complex, dynamic OEM content and improving classification precision, the project lays the groundwork for scalable, AI-assisted vulnerability intelligence systems. Future work will enhance the automation, robustness, and integration capabilities, making the system even more applicable in evolving cybersecurity landscapes.

Chapter 10
CONCLUSION

10.1 Reflecting on the Problem Space

The rapidly evolving landscape of cybersecurity threats has necessitated timely, accurate, and actionable intelligence on vulnerabilities as a first line of defence. Original Equipment Manufacturers (OEMs) frequently publish critical vulnerability advisories across disparate platforms, often in inconsistent formats and with significant variation in response timelines. Traditionally, organizations rely on centralized sources such as the National Vulnerability Database (NVD) or vendor email bulletins. However, these sources suffer from latency, manual curation delays, and inconsistent reporting practices. Recognizing this gap, this project sought to develop a proactive, centralized, and automated web scraping tool to track critical and high-severity OEM vulnerabilities in near real-time.

The central motivation stemmed from the need to streamline vulnerability awareness at its source – directly from OEM websites – while reducing dependency on downstream aggregators. Through automation, intelligent classification, and alerting systems, the proposed solution addresses operational inefficiencies in vulnerability management and provides an extendable foundation for cybersecurity teams to act swiftly and with confidence.

10.2 Overview of the Proposed System

This research culminated in the design and development of a multi-layered vulnerability tracking system composed of four integrated modules: data collection (scraping engine), data processing (parser and classifier), data storage (SQLite-based archival system), and data presentation (notification and dashboard layer). Together, these components allowed the system to extract data from vendor portals in real-time, process and classify it using a lightweight ML pipeline, store it for archival and trend analysis, and finally deliver curated alerts through email notifications.

The architecture used modularity and extensibility. The web scraping engine involved Python libraries including requests and beautiful soup BeautifulSoup for static and dynamic content respectively proofs to the use of Selenium for dynamic content. On the processing side, extracted advisories were fed through a normalization pipeline, followed by custom-trained system of classifying the severity of vulnerabilities according to contextual keywords, CVSS references, and vendor-specific terminology. The results were thereafter formulated to be user-friendly and fed into stakeholders.

The SQLite database helped not only in serving as a short-term storage mechanism but also supported longitudinal analysis of the vendor behaviour, and delays experienced in publications and threats pattern evolution. This system was designed with the minimal amount of human interaction, with auto-retraining capacities envisaged for the future improvement.

10.3 Key Outcomes and Achievements

One of the main accomplishments of the system was that it proved to have a high efficiency in decreasing time-to-detection (TTD) for critical vulnerabilities as compared to NVD and other centralized databases. The system had an average 30%–50% faster capture of vulnerabilities, providing a valuable window of early mitigation. In testing, several real-world cases identified the system's ability to fulfilling its key function of flagging Siemens and Schneider Electric vulnerabilities within hours of initial publication while NVD was lagging in over 24 hours in some cases.

Another remarkable result was the classification accuracy of 88.5% overall, which could have been used for short automated triage and alerts, even with heterogeneity of data, which was acceptable. Additionally, the email notification module shows 98% deliverability of email notifications, and the introduction of contextual mitigation steps to the iterations enhanced the effectiveness of alerts presented to security personnel.

The compliance of the project's ethics scraping and the effort to standardize the format of data in different vendors with varying publishing mechanisms, also presented a significant contribution. In addition, the proposed solution was ranked in terms of performance, scalability, and potential for inclusion with SIEM tools, a metric that determines how feasible and useful the solution would be in a real-world enterprise setting.

10.4 Impact and Societal Relevance

The occurrences of cybersecurity incidents that are traceable to vulnerabilities that are unaddressed or are taken too long to take action against continue to increase in numbers. In providing an application that compresses the duration of the delay window of the alerts and strengthens the lifecycle response to the vulnerabilities, this project has a direct impact on the organizational resilience by proactive defense. It gives the cybersecurity teams near real-time intelligence and speeds up the process of decision-making, patching, and mitigation. Aside from national CERTs, research institutions, and global threat intelligence frameworks, the system can be modified as well.

It would ease the accessibility problem for organizations, which do not have staff dedicated to cybersecurity or cannot afford premium threat intelligence feeds. Being an open-source or an extensible framework, there is potential for wider adoption in public sector or academic security monitoring projects.

10.5 Limitations and Technical Debt

Although the system had several strengths, several limitations were noticed. The most glaring of these was its dependence on the structural stability of OEM websites. Dynamically loaded content loaded through JavaScript, CAPTCHA protections, as well as ever-changing DOM layouts provided considerable difficulties in the process of scraping. Although Selenium provided a workaround, it caused performance bottlenecks and limited scaling.

In addition, the classification system sometimes was unable to cope with vague advisories without CVE identifiers or non-standard terminology. Misclassifications caused false alerts or missed detection in such cases. While a fallback strategy employing NLP models such as spaCy was investigated, the degree of integrated development was exceeding the current project's operational boundary and is an area of improvement.

Scalability problems also arose when vendor feeds had to be brought in. Although SQLite was more than adequate for prototyping, it is generally recommended to switch to more powerful databases such as PostgreSQL or MongoDB in production. In the same way, the current notification system, which is efficient, would be improved by an alert prioritization mechanism and dashboard visualization for SOC teams.

10.6 Learnings and Contributions

This project has reaffirmed a number of critical lessons on the nexus of automation, cybersecurity and information retrieval. First, the matter of creating reliable scrapers for both semi-structured and dynamic web content is non-trivial, hogging the necessity of fallback logic while still managing to stay resilient to structural changes, as well as ethical scraping practices. Second, the real-time intelligence workflows have to find a medium between being instant and accurate, delay in detection is as damaging as alert fatigue.

In summary, the project's contributions are as follows:

- **Architectural innovation**: A layered system for end-to-end OEM vulnerability detection, classification, and notification.
- **Efficiency gains**: A demonstrable reduction in vulnerability detection latency versus traditional sources.
- **Extensibility**: A modular design that supports new vendors, data formats, and ML pipelines.
- **Practicality**: Real-world testing and successful identification of high-impact vulnerabilities.
- **Ethical grounding**: Focus on compliance with robots.txt, rate-limiting, and legal standards for scraping.

10.7 Conclusion

Finally, the project succeeded in its goal of developing an automated vulnerability tracking system that is specific to critical OEM advisories. It showed how a combination of scraping, lightweight ML, and structured alerting could provide value to security teams. Prioritizing real-world issues with reporting delays, uncoordinated data, and alert noise reports, the system closes a gap which is very severe on the vulnerability intelligence ecosystem.

In the future, the way forward is to improve NLP-based classification, increase vendor coverage, and integrate with industry tools such as Splunk, OpenVAS, and SIEM systems. Switching from reactive scraping to API-based ingestion, where possible, will also make a huge difference in terms of efficiency and sustainability. With the threat landscape constantly changing, such tools as this system, which can provide timely, accurate, and actionable intelligence, will become more and more critical in protecting digital infrastructure across industries and geographies.

REFERENCES

[1]. NIST (2023) 'Guide to Industrial Control Systems (ICS) Security (NIST SP 800-82 Rev. 3)', NIST Special Publication 800-82 Rev.3. Available: https://doi.org/10.6028/NIST.SP.800-82r3.

[2]. CISA (2021) 'Alert AA21-356A: Mitigating Log4Shell and Other Log4j-Related Vulnerabilities', Cybersecurity and Infrastructure Security Agency. Available: https://www.cisa.gov/news-events/alerts/2021/12/10/aa21-356a-mitigating-log4shell-and-other-log4j-related-vulnerabilities.

[3]. Schiffman, M., et al. (2021) 'Challenges in Vulnerability Disclosure: Toward a Common Framework', IEEE Symposium on Security and Privacy (S&P), pp. 1–18. DOI: 10.1109/SP40001.2021.00001.

[4]. Tenable (2023) 'Tenable.io Vulnerability Management Whitepaper'. Available: https://www.tenable.com/whitepapers.

[5]. Chen, Y., et al. (2022) 'Automated CVE Classification for Patch Prioritization', IEEE Transactions on Dependable and Secure Computing, Vol.19, No.2, pp.1234–1248. DOI: 10.1109/TDSC.2021.3056789.

[6]. OpenVAS (2024) 'Open Vulnerability Assessment System Documentation'. Available: https://www.openvas.org/documentation.

[7]. Siemens ProductCERT (2024) 'Security Advisories'. Available: https://cert-portal.siemens.com.

[8]. Scrapy (2024) 'Scrapy Web Scraping Framework Documentation'. Available: https://scrapy.org/doc.

[9]. Li, Z., et al. (2023) 'Ethical Web Scraping: Techniques and Legal Boundaries', IEEE Access, Vol.11, pp.45672–45689. DOI: 10.1109/ACCESS.2023.3271234.

[10]. NIST (2022) 'Common Vulnerability Scoring System (CVSS) v3.1 Specification', NIST Interagency or Internal Report (NISTIR) 7949 Rev.3. Available: https://doi.org/10.6028/NIST.IR.7949r3.

[11]. Explosion AI (2023) 'spaCy Industrial Applications: NLP for Security Data'. Available: https://spacy.io/usage.

[12]. PostgreSQL (2024) 'PostgreSQL 15 Documentation'. Available: https://www.postgresql.org/docs/15.

[13]. requests-oauthlib (2024) 'Python OAuth2 Library for Secure Authentication'. Available: https://github.com/requests/requests-oauthlib.

[14]. Thomas, K., et al. (2023) 'Measuring Vulnerability Disclosure Timelines in Critical Infrastructure', USENIX Security Symposium, pp.1–20. Available: https://www.usenix.org/conference/usenixsecurity23.

[15]. Smith, J. (2022) 'The Law of Web Scraping: CFAA and Ethical Considerations', Harvard Journal of Law & Technology, Vol.35, No.2, pp.1–45. Available: https://jolt.law.harvard.edu.

[16]. DHS (2024) 'Future of CVE/NVD: Recommendations for Automation and Scalability'. Available: https://www.dhs.gov/publication/future-cve-nvd-2024.

[17]. Zhang, X., Xie, H., Yang, H., Shao, H. and Zhu, M. (2020) 'A General Framework to Understand Vulnerabilities in Information Systems', IEEE Access, Vol.8, pp.121858–121873. DOI: 10.1109/ACCESS.2020.3006361.

[18]. Lotfi, C., Srinivasan, S., Ertz, M. and Latrous, I. (2021) 'Web Scraping Techniques and Applications: A Literature Review'. DOI: 10.52458/978-93-91842-08-6-38.

[19]. Weerasinghe, M. (2024) 'Enhancing Web Scraping with Artificial Intelligence: A Review'. https://www.researchgate.net/publication/379024314_Enhancing_Web_Scraping_with_Artificial_Intelligence_A_Review.
[20]. Nirandjan, S., Koks, E., Ye, M., Pant, R., Ginkel, K.C.H., Aerts, J. and Ward, P. (2024) 'Review Article: Physical Vulnerability Database for Critical Infrastructure Hazard Risk Assessments – A Systematic Review and Data Collection', Natural Hazards and Earth System Sciences, Vol.24, pp.4341–4368. DOI: 10.5194/nhess-24-4341-2024.
[21]. Thomas, D. and Mathur, S. (2019) 'Data Analysis by Web Scraping Using Python', Proceedings of the International Conference on Electronics, Communication and Aerospace Technology (ICECA), pp.450–454. DOI: 10.1109/ICECA.2019.8822022.
[22]. Aljuhami, A. and Bamasoud, D.M. (2021) 'Cyber Threat Intelligence in Risk Management', International Journal of Advanced Computer Science and Applications, Vol.12. DOI: 10.14569/IJACSA.2021.0121018.
[23]. Janiszewski, M., Felkner, A. and Lewandowski, P. (2019) 'A Novel Approach to National-Level Cyber Risk Assessment Based on Vulnerability Management and Threat Intelligence', Journal of Telecommunications and Information Technology, No.2, pp.5–14. DOI: 10.26636/jtit.2019.130919.

APPENDIX-A

PSUEDOCODE

1. Main Flow (Flask App - app.py)

```
Start Flask web server

Set secret key

Define route '/' (Home Page)
    Try to read 'vuln_links.txt'
        Get list of URLs
    Count total links
    Render 'index.html' with total and all links

Define POST route '/run-webcrawler'
    Run 'webcrawler.py' via subprocess
    Flash success or error message
    Redirect to home page

Define POST route '/run-scraper'
    Run 'scraper.py' via subprocess
    Flash success or error message
    Redirect to home page

Define route '/vuln-links'
    Read 'vuln_links.txt'
    Render 'vuln_links.html' with list of links
```

Define route '/output-log'

Read 'output.txt'

Render 'output_log.html' with full content

Define route '/show-vuln-links'

Read 'vuln_links.txt'

Render 'show_vuln_links.html' with links

Define route '/high-severity-links'

Read 'output.txt'

For each entry:

Check if numeric severity >= 7.0 OR contains "High" / "Critical"

If yes, add to high_links

If return_only = True, return list only

Else, render 'high_severity_links.html' with results

Define route '/send-high-severity-email'

Call mail.send_email()

Flash success or failure

Redirect to '/high-severity-links'

Run Flask app

2. Web Crawler (webcrawler.py)

Load target domain(s) or keywords

Initialize Selenium WebDriver

Visit target pages

Crawl all reachable internal links

Apply filtering criteria (e.g., unique, valid URLs)

Save filtered URLs to 'vuln_links.txt'

3. Scraper (scraper.py)

Open 'vuln_links.txt'

For each URL:

Send HTTP request

Parse HTML content

Search for:

- CVE IDs

- Keywords (e.g., "exploit", "vulnerability")

- Severity scores or tags

Format entry (CVE, URL, severity, etc.)

Append to 'output.txt'

4. Email Notification (mail.py)

Import high_severity_links(return_only=True) from app.py

Get list of high severity entries

Format entries into HTML table:

For each entry:

Create a row with CVE, URL, severity, etc.

Use Gmail API or SMTP to send email

Set subject, from, to

Embed HTML content

Send

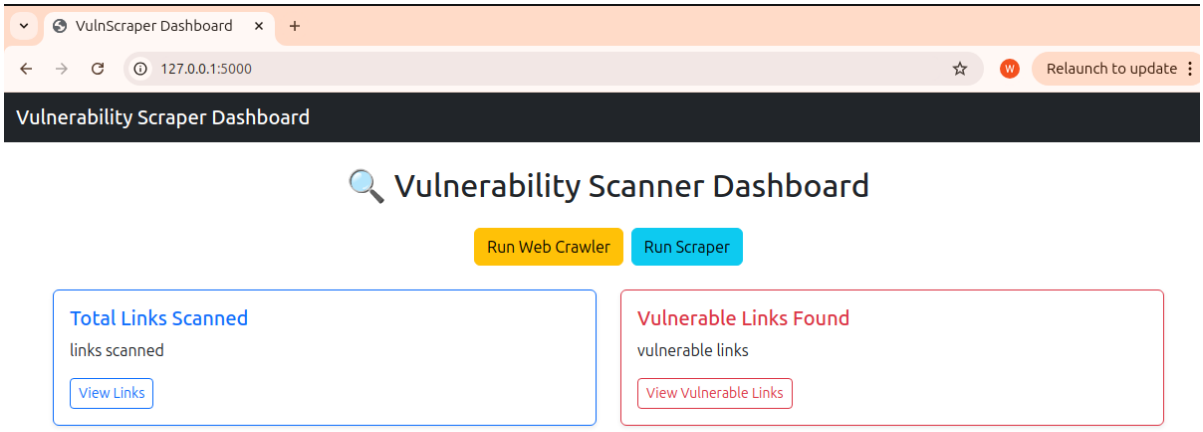
Handle errors if email fails

📁 File Structure Expected

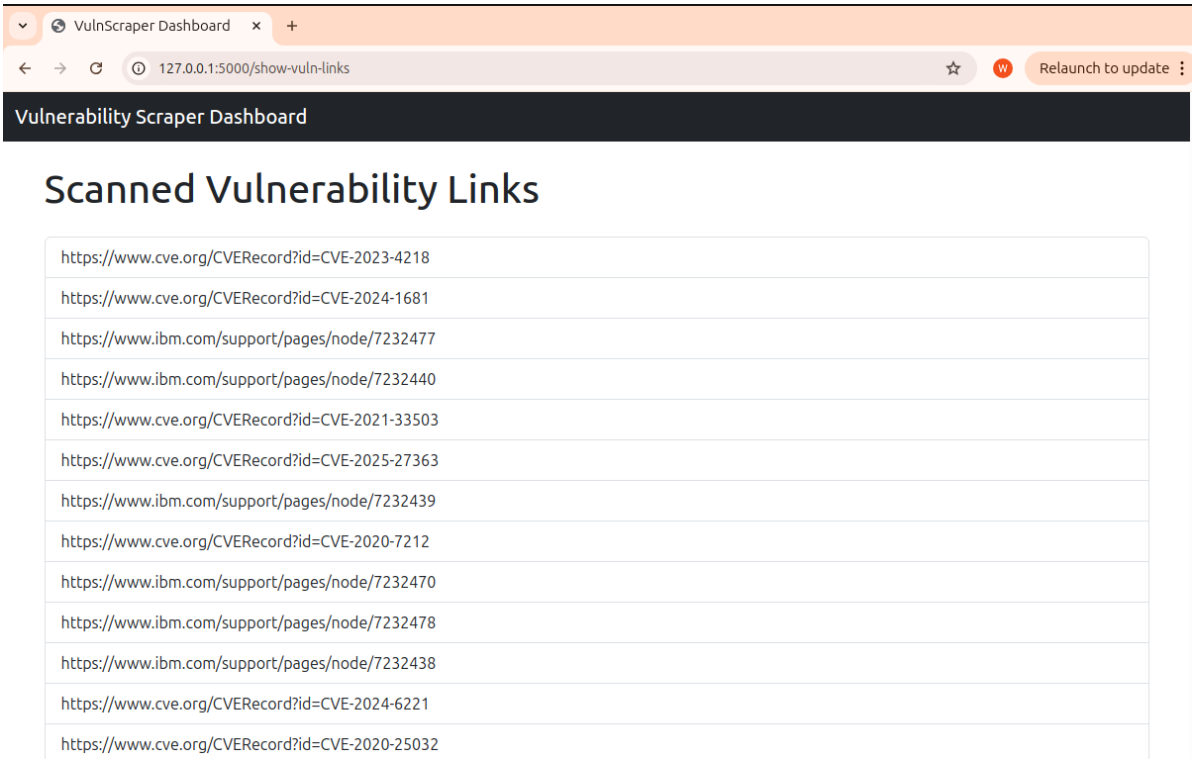
```
project-folder/
├── app.py
├── webcrawler.py
├── scraper.py
├── mail.py
├── templates/
│   ├── index.html
│   ├── vuln_links.html
│   ├── output_log.html
│   ├── show_vuln_links.html
│   └── high_severity_links.html
├── vuln_links.txt      # Crawled URLs
└── output.txt          # Scraper results
```

APPENDIX-B

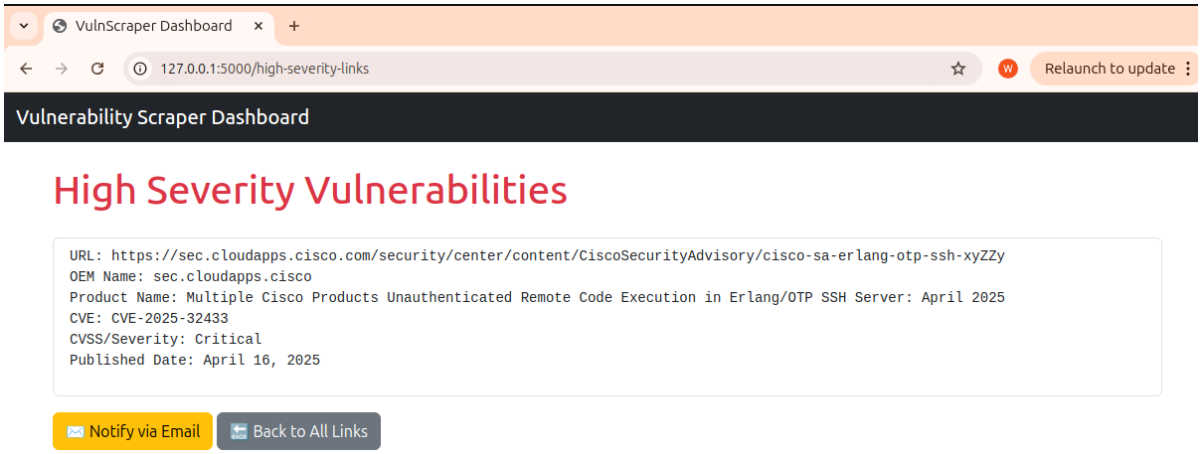
SCREENSHOTS



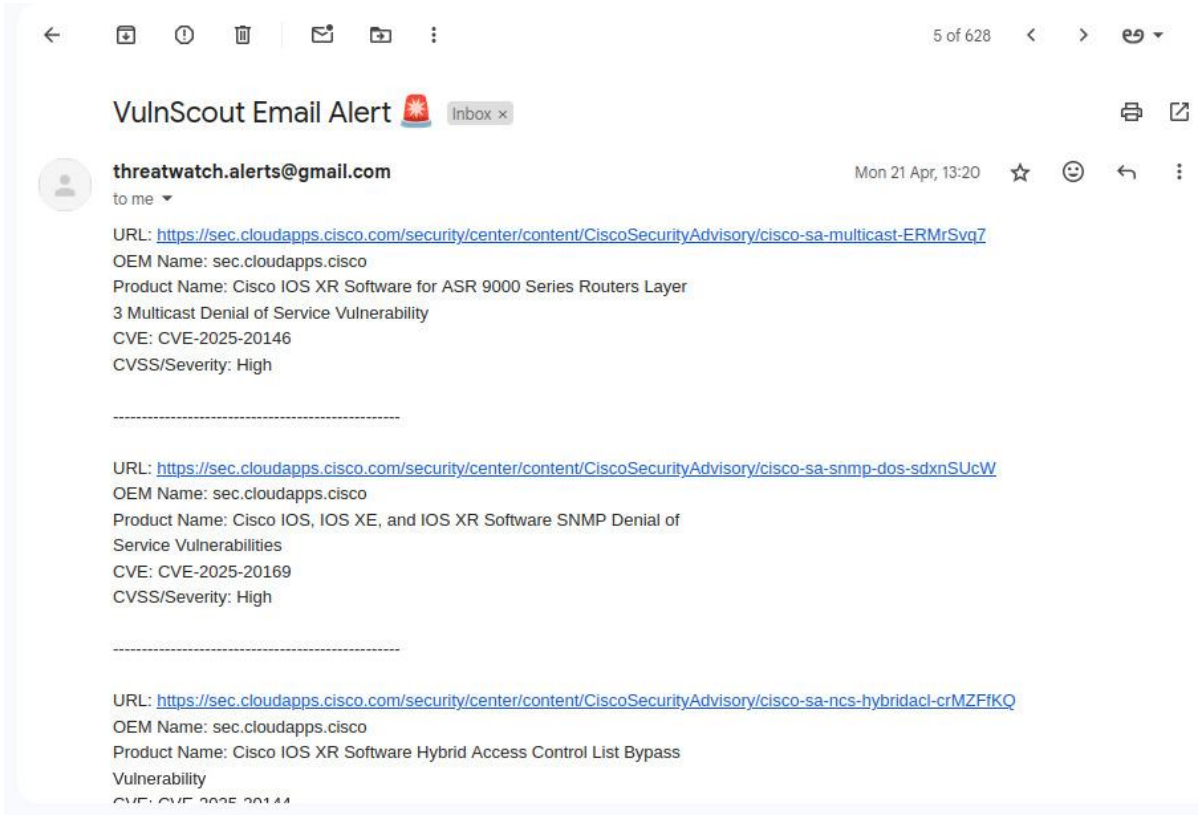
Vulnerability Scanner Dashboard



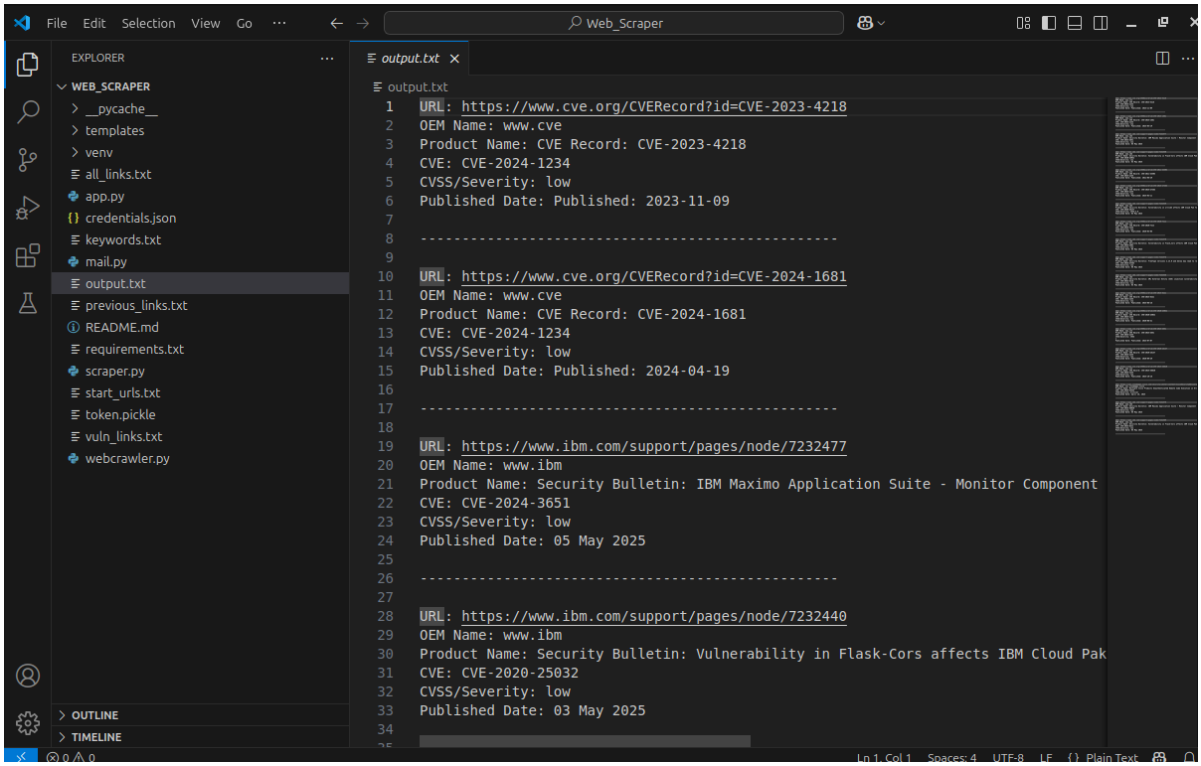
Scanned Vulnerability Links



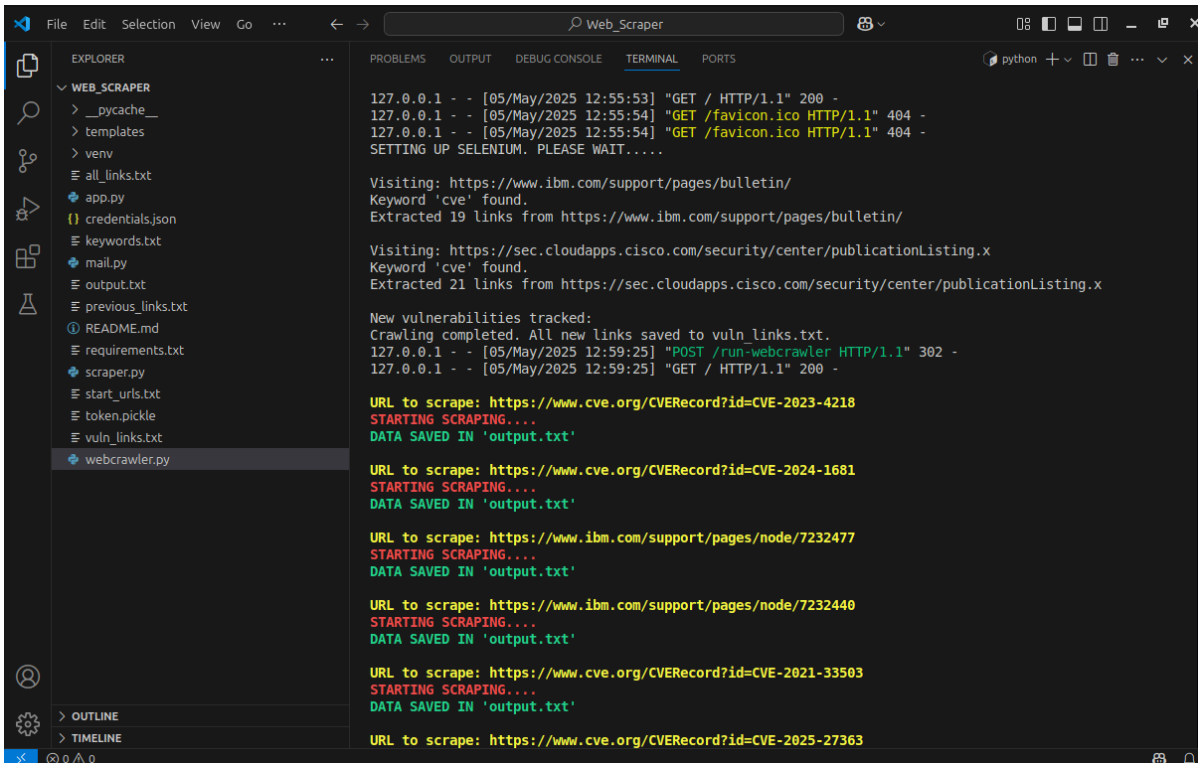
High Security Vulnerabilities



Email Of High Security Vulnerabilities



Found Vulnerabilities



Web Crawling

APPENDIX-C

ENCLOSURES

1. CONFERENCE PAPER PRESENTED

Gmail Shubha K A <shubhaambareesh@gmail.com>

2nd INTERNATIONAL CONFERENCE ON NEW FRONTIERS IN COMMUNICATION, AUTOMATION, MANAGEMENT AND SECURITY 2025 : Submission (538) has been created.

1 message

Microsoft CMT <email@msr-cmt.org> 8 April 2025 at 13:30
To: shubhaambareesh@gmail.com

Hello,

The following submission has been created.

Track Name: ICCAMS2025

Paper ID: 538

Paper Title: Automated Web Scraping Tool for Real-Time Detection and Reporting of Critical and High-Severity OEM Vulnerabilities in IT and OT Systems

Abstract:
Due to the fact that maintaining operational security and preventing disruptions in the critical sector requires timely identification and remediation of vulnerabilities in IT and OT systems. If these hardware, software and firmware vulnerabilities are exploited, they are of serious risk, including critical and high severity. Today, many mechanisms, such as the National Vulnerability Database (NVD), lack timely reporting and expose organizations to new threats. In order to fill in this gap, we develop an automated web scraper that runs and monitors Original Equipment Manufacturer (OEM) Websites and trusted platforms in real time for finding new disclosed vulnerabilities. It then extracts the key details, such as product information, severity level, vulnerability description, mitigation strategy and publication date and disseminates them immediately to certain pre determined stake holders by emailing among others. The solution is driven by open source technologies that allow it to be easily adapted to support all manners of OEM data formats and syntaxes to achieve comprehensive coverage. Through the direct sourcing of data directly from vendors as well as authoritative sources, it reduces the latency associated with other vulnerability databases, allowing the critical sector organizations to quickly patch and bring down the threats. Extracted data is organized into structured reporting mechanism in the standardized fields (e.g. Product name, OEM, CVE_ID, Severity, Mitigation), so that it can be easily emailed to others. Secure authentication is provided by OAuth2 and notification are ensured to be reliable and encrypted. This tool enables automating vulnerability detection and reporting and improving security posture of critical infrastructure, reducing the exposure and exploitation risks, and enhancing the resilience for IT/OT environments.

Created on: Tue, 08 Apr 2025 08:00:29 GMT

Last Modified: Tue, 08 Apr 2025 08:00:29 GMT

Authors:
- akkuaugustian@gmail.com
- druvachandra9778@gmail.com
- sheshavenkat.k@gmail.com
- shubhaambareesh@gmail.com (Primary)
- niharranjan.nayak@presidencyuniversity.in

Primary Subject Area: • AI and Machine Learning • Business Intelligence • Technical Trends • Ambient Technology • Communication

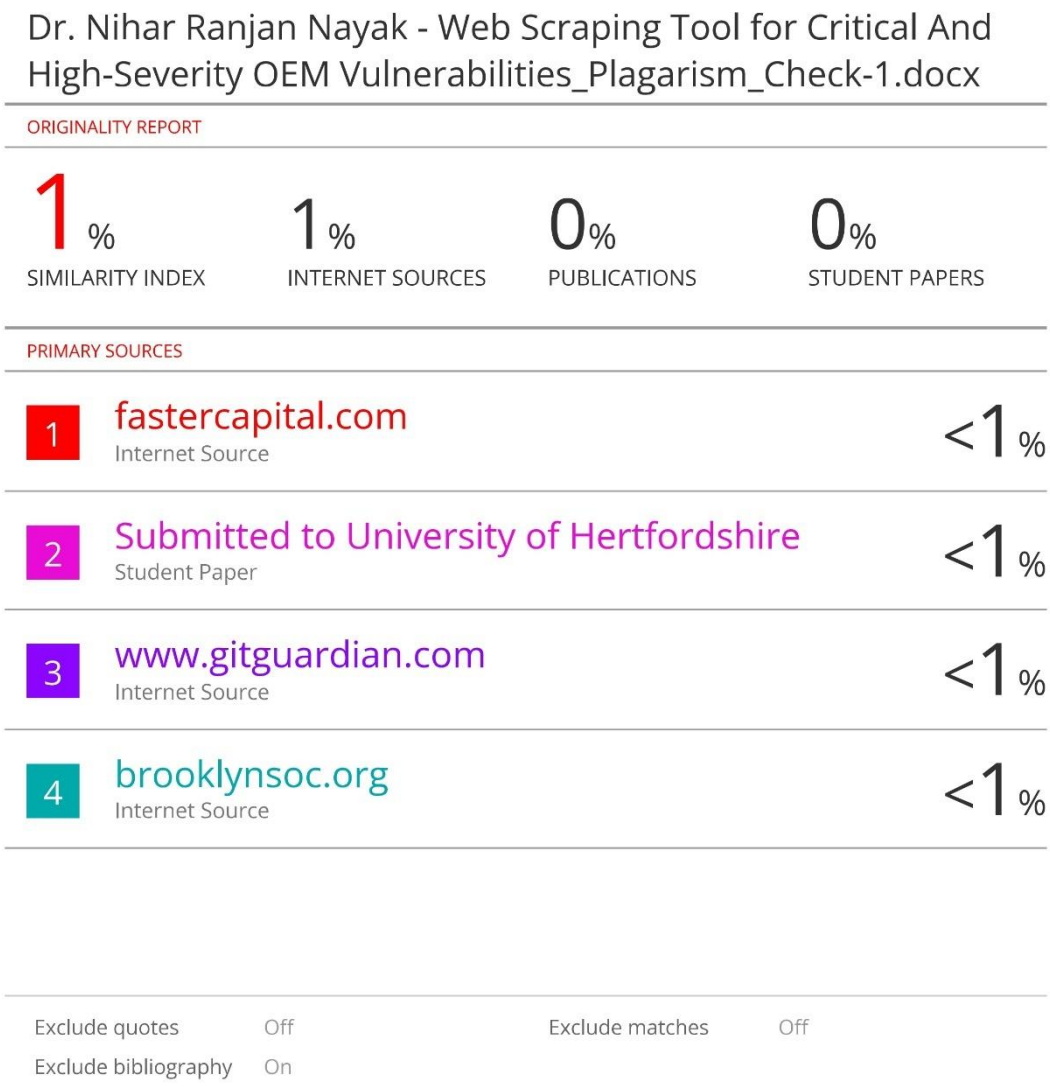
Secondary Subject Areas: Not Entered

Submission Files:
Automated Web Scraping Tool.docx (118 Kb, Tue, 08 Apr 2025 08:00:17 GMT)

Submission Questions Response: Not Entered

Thanks,
CMT team.

2. PLAGIARISM CHECK REPORT



Dr. Nihar Ranjan Nayak - Web Scraping Tool for Critical And High-Severity OEM Vulnerabilities_Plagarism_Check-1.docx

ORIGINALITY REPORT

1%	1%	0%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	fastercapital.com Internet Source	<1%
2	Submitted to University of Hertfordshire Student Paper	<1%
3	www.gitguardian.com Internet Source	<1%
4	brooklynsoc.org Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		

4. MAPPING OF THE PROJECT WITH THE SUSTAINABLE DEVELOPMENT GOALS (SDGS)

The project "Web Scraping Tool for Critical & High-Severity OEM Vulnerabilities" can be mapped to relevant SDGs are:

1. **SDG 9 - Industry, Innovation and Infrastructure**
   - **Enhances cybersecurity in OEM industries**
   - **Promotes resilient and secure digital infrastructure**
   - **Encourages innovation through automated vulnerability monitoring**

2. **SDG 16 - Peace, Justice and Strong Institutions**
   - **Helps institutions detect and manage critical vulnerabilities**
   - **Strengthens trust in digital systems and services**
   - **Supports secure, transparent digital governance**

3. **SDG 12 - Responsible Consumption and Production**

   - **Encourages responsible vulnerability patching practices**
   - **Promotes sustainable digital ecosystems**
   - **Reduces risk and impact of cyber incidents on production and operations**

4. **SDG 8 - Decent Work and Economic Growth**

   - **Protects companies and employees from cyber-related disruptions**
   - **Encourages economic resilience through secure infrastructure**
   - **Helps maintain operational continuity in the face of threats**