Secure Scan: Live Monitoring of Critical IT Vulnerabilities

Dr. Priyanka Kadam[1], Swaroop Ingle[2], Harsh Gujalwar[3], Himanshu Khandelwal[4], Shrirudra Khanderay[5]
[1]*Project Guide, Vishwakarma Institute of Technology*
[2,3,4,5]*Computer Engineering Vishwakarma Institute of Technology*

Abstract— **When vulnerabilities are not disclosed and false alerts are raised in various critical sector environments; it can lead to loss of trust in security systems. This paper introduces a system which integrates real-time vulnerability crawling from OEM security advisories and technology stack detection on the target or user's system. The system identifies the software that an organization is using, it keeps continuously downloading new vulnerability information, and compares it with the software stack that it has detected for the target organization and only displays alerts that are relevant to it. It can be demonstrated by a fully functional pipeline that fetches critical CVEs from various sources; identifies vulnerable stacks for target websites; matches the vulnerabilities using a CPE mapping service; and shows the results in a service running on MongoDB. The system filters out irrelevant data, filters out false-positives and notifies the user of critical data just in his environment (faster response time). This is not only a scalable framework for vulnerability intelligence in IT infrastructure but also a way to improve real-world things' proactive security monitoring.**

Keywords-- vulnerability monitoring, OEM scraping, technology stack detection, real-time alerting, IT security

I. INTRODUCTION

Over the years, cybersecurity has emerged as the greatest problem in today's digital world. Interconnected IT systems are critical infrastructures, such as healthcare, finance, energy and government, and are highly susceptible to cyberattacks. The presence of software and hardware vulnerabilities in these environments can lead to operational downtime, financial losses or significant risk to human safety. The number of Common Vulnerabilities and Exposures (CVEs) has increased significantly over the last few years, and attacks are expanding their surface area and even the velocity at which vulnerabilities are becoming threats.

Most organisations use the National Vulnerability Database (NVD) or a similar repository to keep track of newly reported vulnerabilities. NVD is a useful reference, but has significant flaws. The most common one is latency; vulnerabilities reported by the Original Equipment Manufacturer (OEM) are usually reported on the NVD sites several days after the report. A delay in the deployment of systems that are connected to the internet gives adversaries the time they need to make vulnerabilities into a high-impact exploit. Otherwise, NVD gives information in bulk, with vulnerabilities on organisations that may not be deployed in their technology stack. These may cause ‘alert fatigue' and latency in the reaction time of security teams to act on a specific CVE due to the time they have to spend on analyzing whether the CVE impacts their infrastructure or not.

One of the most salient inadequacies of today's feeds is that they do not provide context. While current systems allocate ratings to the vulnerabilities and some generic metadata, they typically don't correlate the vulnerability with the deployment profile of the organisation. A company running Apache HTTP Server might receive equal-attention alerts for vulnerabilities in other systems, such as Microsoft SQL Server or Cisco IOS, which can result in an abundance of noise. This is not only time-consuming and resource-intensive in terms of information but also the risk of missing out on vital vulnerabilities.

To overcome these drawbacks, we introduce SecureScan, a proactive vulnerability intelligence framework, which brings together timeliness,

contextual relevance and actionable insights. It is continuously scanning for vulnerabilities from trusted OEMs and NVDs, as it also fingerprints the technology stacks of target systems. SecureScan correlates the identified tech stacks with real-time disclosures, creating real-time and relevant alerts.

The novelty of the approach is that it is a method of connecting raw disclosure with actionable information. SecureScan provides targeted alerts, unlike traditional monitoring, which treats every vulnerability the same. This not only helps organisations to patch faster, but also helps them to decide which mitigation strategies to take. By doing that, the system helps to create a more proactive cybersecurity stance, in line with the increasing industry focus on early detection and fast response.

II. LITERATURE REVIEW

Software systems have increased in number and complexity as have cyberattacks, and, in the past decade, so has the need for Vulnerability Management. However, according to some research, there are numerous vulnerabilities that are being discovered annually, and if companies, especially in critical industries, could not keep track of them through a minor lag time, then they could be faced with a disrupted business. Vulnerability management has come a long way in the last 10 years, thanks to the explosion of software systems and the sophistication of attacks. There are a huge number of vulnerabilities discovered every year, sometimes tens of thousands; it is impossible to keep up with these vulnerabilities by hand, especially for organisations in critical industries where a delayed discovery can have direct (and negative) implications for operational continuity. Leverett et al. showed that vulnerability disclosures adhere to predictable statistical patterns and that models capable of predicting vulnerability disclosure trends are able to predict disclosure patterns within reasonable bounds, which highlights the importance of having automated systems that are able to ingest and process vulnerability disclosures in real time [1]. This has given the idea to further investigate if time-series and deep learning models can be used to predict vulnerability surges for individual platforms. For example, Gencer and Bourekba [2] used vulnerability data for Android to apply ARIMA and LSTM models to demonstrate that automated vulnerability prediction tools can be useful for patch planning and resource allocation. These works together indicate that vulnerability data is voluminous and continually changing, and thus, the need for an automated system that can ingest them in a timely fashion.

Another common theme found in literature is the latency of vulnerability databases such as the National Vulnerability Database (NVD). Ruohonen et al. conducted a thorough empirical study and found that there are a number of ways that the NVD entries are delayed from the vendor (OEM) advisories, sometimes even hours to days [3]. This lag time is important as security researchers generally use NVD as a cornerstone of their security initiatives, which can leave attackers ahead of them for exploiting vulnerabilities before they can even find out about them. In a study focused on forecasting security vulnerabilities, Yasasin et al. also emphasize the critical need for systems that can retrieve and correlate OEM-level advisories for enterprises to plan timely patching strategies, further hindering the challenge of finding vulnerabilities [4]. Together, these results validate the design decision at SecureScan to eventually be complementing NVD data with OEM scraping.

Contextual relevance is the second key gap in present vulnerability alerting ecosystems; in addition to timeliness. The vast majority of raw CVE feeds list vulnerabilities for thousands of products, many of which are not relevant to any particular organisation. WhatWeb, Wappalyzer and BlindElephant are studied fingerprinting tools to detect what are the real technologies a running website or application is using. One of the most recent and thorough studies was conducted by Kondracki et al., which demonstrated that fingerprinting is viable but requires ability to obfuscate, camouflage the version, and have inconsistent response patterns [5]. Fingerprinting, however, when used in conjunction with structured analysis, can be a useful tool to restrict the number of vulnerabilities that need to be considered in a particular environment. This observation matches closely SecureScan's methodology, which involves detecting the technology stack first before vulnerability matching is performed.

To reliably identify vulnerabilities, the components must have structured identifiers. While imperfect, the

Common Platform Enumeration (CPE) system is recognized in the literature as the most common way of associating CVE entries with products. To improve the mapping accuracy, Wareus et al. suggested the use of machine-learning and named-entity-recognition (NER) to automatically assign CPE labels to CVE descriptions, which greatly reduces human effort [6]. Likewise, Shi et al. created a model based on a knowledge graph to correlate the entities of CWE, CPE, and CVE, which is able to minimize the association errors and enhance the precision of correlation [7]. These works emphasize that matching based on keywords is insufficient and structured metadata, supplemented by metadata enrichment based on machine learning, is required to get accurate results in vulnerability correlation.

Another space that is also relevant to SecureScan is the vulnerability prioritisation and triage space. There have been many studies that have raised the question that CVSS scores don't represent the actual exploitation potential in the wild. Sabottke et al. found that social media and signals from the exploit markets can be used to predict which vulnerabilities are likely to be exploited in the wild [8], and Nayak et al. proved that machine-learning models that are trained using exploit data and environmental conditions are much more effective at prioritisation than traditional CVSS-based prioritisation [9]. These works indicate that future implementations of systems such as SecureScan may include extra prioritisation layers beyond relevance, to rank items for taking action.

In recent years, large language models (LLMs) have been a focus of research for vulnerability analysis and remediation assistance. One of the earliest rigorous studies of using LLMs for vulnerability triage, by Zhang et al., suggests that models, such as GPT-4, are capable of summarizing and explaining CVEs very well, but hallucinating requires knowledge of authoritative data [10]. Likewise, Pang et al. showed that LLMs can help with configuration fixes and patching advice with accurate vulnerability metadata [11]. These results confirm that SecureScan's implementation of an LLM-powered chatbot is appropriate, and underscores the importance of the proper structure of the input, using validated CVE data.

Last but not least, problems associated with OEM advisory aggregation are highlighted. Vendors post advisories in an HTML page, an RSS feed, a mailing list, a PDF, and their own security portal—rather than a standardized machine-readable file. A systematic investigation of IoT vulnerability sources by Rytel et al. found that one of the biggest challenges in automating early vulnerability detection is the reporting format inconsistencies across the OEMs [12]. The industry also suggests that combining OEM advisories with centralised databases like NVD will provide the maximum coverage, which SecureScan does by including a modular scraping feature.

Combined, these studies highlight a lack of understanding: although various themes have been studied individually, there is a gap in the literature as few studies connect these threads in an integrated, operational pipeline. SecureScan fills this void by providing real-time vulnerability acquisition, stack detection, structured correlation, and LLM-driven advisory generation all within one system.

III. DESIGN METHODOLOGY

This Paper introduces the system developed to be an end-to-end automated pipeline of vulnerability analysis, to identify high risk vulnerabilities relevant to an organisation's actual technology stack. The solution includes vulnerability data gathered, technology detection, correlation of all the data, LLM-driven recommendations and an interactive user interface. The methodology is divided into the following main phases as follows, to ensure modularity and clarity, requirement analysis, designing system architecture, implementing backend services, integrating the stack-detection module and the chatbot advisory model, UI development and testing & validation.

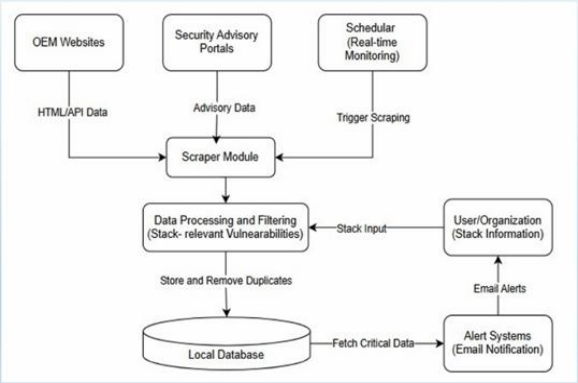

Figure 1 Flowchart of the System

Requirement Analysis

To set up a viable and scalable approach for minimizing false positives in vulnerability reporting alerts in cybersecurity, a detailed requirement analysis was performed. Traditional systems provide too many CVEs, too many of which may not necessarily apply to the user's environment. The thing was, to create a system that would be able to keep track of the real-time vulnerabilities and, based on the user's actual tech stack, filter out only the relevant and actionable alerts.

Based on this requirement, the basic system modules were identified:

1. Data Collection Module

Responsible for pulling vulnerability information from NVD and, in more comprehensive versions, OEM websites and feeds.

Testing the stack of the technologies.

2. Detection module for technology stack.

Fingerprints the technologies of a desired site or app, like Wappalyzer.

3. Correlation Engine

Relates technologies to vulnerabilities and only retrieves high impact, relevant CVEs.

4. Chatbot Advisory System

Can read the output and understand mismatches or errors and give remediation guidance using an LLM

5. Frontend Dashboard

A flexible, React-driven UI for URL input, visualisation and interaction with the chatbot.

6. Database Layer

Structured Storage System (MongoDB) for Caching CVEs and Querying them efficiently.

These requirements are at the core of the SecureScan, and they give users a solution that meets their requirements as well as the security team's workflows.

System Architecture Design

Architecture of SecureScan has been designed to be modular and service oriented to be easily scaleable and understandable as processing of data. It starts by the user entering a target URL from the dashboard. This triggers an API call to the backend FastAPI service which orchestrates the below sequence:

1. Vulnerability Fetching Pipeline

The back-end makes requests to the NVD API (and subsequently OEM advisory pages) to obtain the latest list of high severity vulnerabilities. The data is normalised and inserted into MongoDB.

2. Stack Fingerprinting Module

Using a Wappalyzer-driven library, the backend extracts the target website's technologies such as web frameworks, server configurations, CMS platforms, and JavaScript libraries.

3. Correlation Engine

The backend matches the detected stack to the vulnerabilities found in the MongoDB database. Only results with overlapping keywords, vendor names, product identifiers and/or version ranges are returned.

4. Chatbot Recommendation Unit

This vulnerability list or system errors (e.g. no stack detected, missing information, unmatched data) are then passed on to the LLM integrated via the backend. The chatbot analyses the context and provides information on mitigation or further action.

5. Frontend Display

The cleaned output, alongside with the chatbot recommendations, is sent back to the React UI for user interaction.

Architectures concentrate on clarity, reusability of components and structured data flow from detection to recommendation. It can also help with smooth future integration of scheduler automation, alert emails, OEM scraping, etc. without any architectural redesign.

Backend Implementation and Data Processing

The back-end is the main controller responsible for managing all the parts of the system. The FastAPI is implemented for the backend because of its speed, async nature and clean declarative structure. On receiving the request, the backend does:

1. Data Fetching Logic

Pulls recent CVEs from the National Vulnerability Database using REST API. To ensure relevance and optimized retrieval, parameters like severity levels, date range, and keywords are used.

2. MongoDB Storage & Normalisation

The CVE entries retrieved are normalized and put into a consistent schema and stored in MongoDB. The step improves the searchability, increases the matching speed.

3. Stack Detection Integration

The back-end invokes the fingerprinting tool and examines the technology elements of the domain being fingerprinted. Returned data is split into product names, versions and categories.

4. Matching Logic Execution

The back end detects vulnerabilities that apply to the technologies found through keyword matching, vendor-product association and basic component mapping. This is the baseline for Vulnerability Reporting that can be done on a targeted basis.

5. Error Handling Pipeline

Various exceptions are handled, including bad URL links, missing vulnerability entries, failed fingerprinting attempts etc., and sent to the chatbot module for actionable feedback.

The backend is structured with modular utility functions and asynchronous handling of requests for efficiency and scalability.

Technology Stack Detection

The stack detection module is one of the most important ones because it deals with the interpretation of technologies of a target website. The process includes:

The steps involve sending HTTP requests to the target domain.These steps include sending HTTP requests to the target domain.

• Parsing response headers, cookies, script headers, HTML

• The ability to recognize fingerprints corresponding to known patterns in the detection library.

• Returning of structured JSON of frameworks, libraries, versions, servers and platforms

This module is crucial as it sets the context that affects which vulnerabilities are relevant to the user.

Vulnerability Correlation Engine

When the fingerprint technologies become available, the back-end runs a multi-step correlation process:

Technology identifiers are extracted from the text. The text is scanned for extracting technology identifiers. Vendors, product, version and keywords are extracted and normalized.

1. Search Query Construction

Conditions like: are used to write queries for MongoDB.

- Keyword matches
- Vendor-product associations
- Severity filters
- Date thresholds

2. Relevance Ranking

If more than one vulnerability exists in a stack, they are sorted by severity, exploitability and publication date.

To ensure the accuracy of data, certain rows were filtered for False Positives.

The number of non-relevant matches generated by the hits of those keywords is reduced by checking context fields.

The result is a list of vulnerabilities that is clear, concise, relevant, and really describes the vulnerabilities found in the detected stack.

Chatbot Advisory Module

SecureScan also includes a chatbot-based advisory subsystem to help organisations with limited cybersecurity skills. This LLM-powered component:

- Interprets vulnerability descriptions
- Reads error messages or unmatched cases
- Identifies responses that can be taken by humans to remediate the situation
- Provides recommendations for mitigation solutions, including patching, configuration changes or upgrade paths
- Offers easy-to-understand description of CVEs for non-technical users

This converts raw vulnerability information to intelligence and makes SecureScan more practical for both students and small teams and professionals.

Frontend Dashboard & User Interaction

React-based user interface allows for simple interaction with the system. The dashboard includes:

- URL input module

  The output display for the technology stack. The output display of the technology stack.

  A vulnerability table containing severity indicators is provided.

  There is a suggested fixes panel to chat with the bot.
- Clean UI with cloud-like design, responsive design

All interactions are real-time, via REST API calls to the back end. This makes SecureScan a contemporary vulnerability analysis platform that can be accessed by any browser.

Testing and Validation

A large number of tests were carried out to ensure accuracy, robustness and usability. The tests included:

- Different domain URLs and URLs that use different technology stacks.
- Companies that use Apache, Nginx, WordPress, React, Node.js, Flask, etc.
- The correctness of the matching of the CVE has

been verified.

- Error Handling validation and chatbot suggestions validation.
- Covers testing using interrupted requests, invalid URLs and unknown stacks.
- Stress tests MongoDB with large amounts of CVE data.

The outcomes demonstrated that SecureScan regularly provided pertinent vulnerabilities in high accuracy and the chatbot successfully assisted users in finding vulnerabilities.

IV. RESULTS AND ANALYSIS

The proposed vulnerability assessment system was tested on several real-world websites to assess its accurateness in detecting vulnerabilities, efficiency in response and quality of the vulnerabilities detected. In the initial testing period, vulnerabilities from the technology stack of each website were identified and successfully detected by the system. The detection output is summarized in Figure 2, showing that the system can successfully count the stack specific weaknesses according to the pre-defined of vulnerabilities which are stored in MongoDB database. The back-end always interpreted the structure of the webpages, the structure of the response headers, embedded scripts and metadata, keeping the analysis in line with current security configurations.

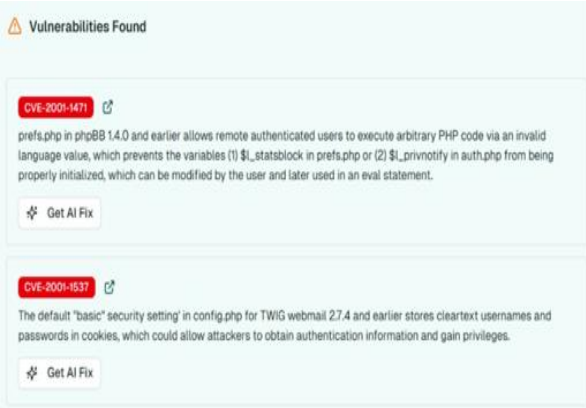

Figure 2 Detected Vulnerabilities based on the tech stack for the Bollyflix site

The system not only identified weaknesses, but also incorporated an interpretation module powered by AI to improve usability for both technical and non-technical users.

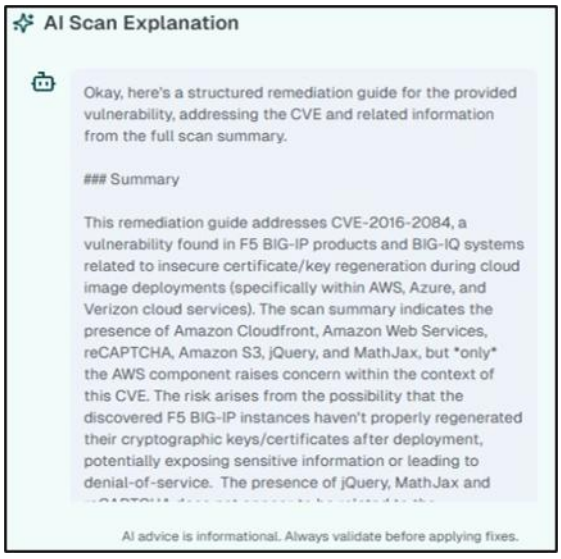

Figure 3 AI-generated explanations and remediation guidance for the detected vulnerabilities.

The LLM based assistant gave a clear explanation of the vulnerabilities found and actionable remediation advice as shown in figure 3. This greatly enhanced the value of the results, making them more easily understood by users who are not deeply involved in cyber security, and put each issue in the context of the overall security posture of the target website.

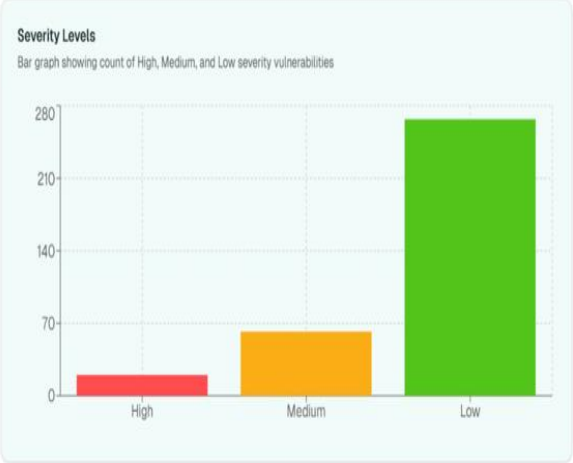

Figure 4 Severity-level distribution visualised using bar-chart analysis.

The system also produced graphical summaries of the levels of vulnerability found with further analysis of the severity of the problems detected. As seen in Figure 4, the severity-level distribution shows the percentages of the test set for high, medium and low severity vulnerabilities. These visual analytics were essential to the users' ability to prioritize remediation efforts,

particularly when there were several layers of risk in the scanned environment.

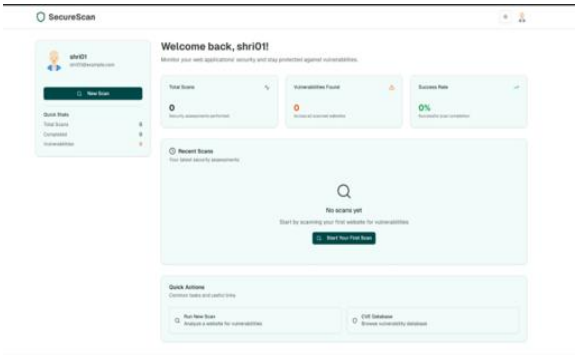

Figure 5 Dashboard of the SecureScan

The system's user interface also was effective in providing the results in structured and accessible format. The dashboard (shown in Figure 5) combines detailed listings of vulnerabilities, their severity, and graphical information into one view. The streamlined presentation, coupled with email-based reporting allowed users to conveniently access and store their assessment results. Overall, the evaluation shows that the system provides rapid, accurate and understandable vulnerability analyses. A modular architecture and extensible rule-based framework give it the ability to scale and adapt to be a constant web security monitor as new threats arise.

REFERENCES

[1] É. Leverett, M. Rhode, and A. Wedgbury, "Vulnerability Forecasting: In Theory and Practice," *arXiv preprint*, 2020.

[2] K. Gencer and F. Bourekba, "Time Series Forecast Modelling of Vulnerabilities in the Android Operating System Using ARIMA and Deep Learning Methods," *Sustainable Computing*, 2021.

[3] J. Ruohonen, "A Look at the Time Delays in CVSS Vulnerability Scoring," *Computers & Security*, 2020.

[4] E. Yasasin, J. Prester, G. Wagner, and G. Schryen, "Forecasting IT Security Vulnerabilities: An Empirical Analysis," *Computers & Security*, vol. 98, 2020.

[5] B. Kondracki et al., "Smudged Fingerprints: Characterising and Improving Web Application Fingerprinting," *USENIX Security Symposium*, 2024.

[6] E. Wåreus et al., "Automated CPE Labelling of CVE Summaries with Machine Learning," *Journal of Cybersecurity*, 2022.

[7] Z. Shi et al., "Uncovering CWE–CVE–CPE Relations via Threat Knowledge Graphs," *ACM Transactions on Privacy and Security*, 2023.

[8] C. Sabottke, O. Suciu, and T. Dumitraș, "Vulnerability Warning Aggregation and Prioritisation with Social Media," *USENIX Security Symposium*, 2015.

[9] A. Nayak et al., "Improving Vulnerability Prioritisation Using Machine Learning," *IEEE Access*, 2021.

[10] C. Zhang et al., "Evaluating Large Language Models for Vulnerability Analysis," *arXiv preprint*, 2023.

[11] R. Pang et al., "Large Language Models for Cybersecurity: Capabilities and Limitations," *arXiv preprint*, 2024.

[12] M. Rytel et al., "Towards a Safer IoT: A Survey on Vulnerability Sources and Disclosure Practices," *Sensors*, vol. 22, no. 16, 2022.