A Review on Multi Target Web Vulnerability Scanner: SamSec

Vaishnavi[1], Janvi Tyagi[2], Nisha Panchal[3], Shruti Jindal[4]
[1,2] *Student, Department of Computer Science and Engineering (Cyber Security) Panipat Institute of Engineering and Technology, Samalkha, Panipat, Haryana, India*
[3,4] *Department of Computer Science and Engineering (Cyber Security) Panipat Institute of Engineering and Technology, Samalkha, Panipat, Haryana, India*

***Abstract*—The automated vulnerability scanning is now a part and parcel of contemporary cybersecurity landscape due to the quick evolution of the web applications and dynamic environment surrounding the hosting location and the quick evolution of the exploits, which requires the development of efficient and effective vulnerability scanners. Despite its ability to analyze the majority of the features in details, SANS It is resource intensive; consequently, the idea of manual scanning cannot be used to provide a continuous and integrated security scan required by the latest DevSecOps pipeline and the deployment of massive infrastructures. In addition, the enterprise systems that are available, such as OWASP Amass, Nmap, and Burp Suite Professional, may be able to provide deep- inspection capabilities, but also have severe limitations in the shape of high latency, high memory footprint, or lack of real interoperability of the automated processes. The provided paper provides a detailed and structured coverage of the contemporary, modularized automated scanning tools (Subfinder, HTTPX, Naabu, Nuclei) and presents a proposal of a simple, fast, and complete-dock reconnaissance provision tool, which is applied in vulnerability scanning drills and named SamSec. Unlike the novel scans of havocs founded on LLM (Large Language Model), which demand a high calculation intensity, and that yield non-deterministic outcomes, SamSec will support the principles of deterministic tooling and the microservice structure based on optimal criteria in velocity and dependability. Furthermore, we are talking about the problem of engineering, which deals with the alignment of tools across platforms, standardizing the differences between the data of dissimilar JSON schemes, and managing the long-lasting asynchronous jobs. And finally, we will know how the future of correlating and ongoing scanning vulnerabilities with AI advancements and distributed scanning models will be.**

I. INTRODUCTION

Over the past decade, web application security has experienced a major shift in focus. As more and more organizations transition to using microservices and cloud-native environments, the potential attack surface of an enterprise has ballooned (from a few monolithic servers to thousands of dynamic endpoints, application programming interfaces (APIs), and sub- domains). Based on OWASP's (Open Worldwide Application Security Projec)Top 10 trends (ranks) for 2025, outdated softwares /components and misconfiguration are most common factors in many breaches. The precarious nature of the security industry, in addition to the volume of infrastructure that security teams have to maintain, can lead to errors or missing elements during the monitoring process.

The web reconnaissance stage of a penetration test is a crucial first step in the testing process; however, web reconnaissance is usually performed manually and uses a combination of ad-hoc scripts, command line utilities, etc. The problem with the manual approach to reconnaissance is that it has three significant issues:

1. Inconsistency. Due to the human element in creating the scripts, testers can either miss a check or skip checks entirely, which could result in missing transient assets.
2. Latency. Manual info gathering of multiple sub-domains is too time consuming to be practical for most organizations.
3. Scalability. Using Continuous Integration and Continuous Delivery (CI/CD) for patching and deploying applications has eliminated the need for manual checks to see which assets are available.

The security industry has witnessed the rapid growth of automated scanning tools as responses to challenges faced. Early examples of automated scanning solutions, such as Nessus and Nikto were effective at providing one-stop shopping when it came to automated scanning; however, they tended to

be fairly slow and were often time consuming and difficult to integrate into custom-built workflows/pipelines. Currently, the trend is toward distributed or task- oriented tools that are smaller in size and have been built with the “Unix Philosophy” approach that promotes single purpose applications, such as Subfinder for enumeration purposes and Naabu for determining which ports are open on network hardware.

Even though the development and use of these various security applications have helped to address many of the issues associated with scanning, integrating them into a unified scanning solution presents additional engineering problems, most notably, normalizing input and output data formats from the distinct applications and ensuring compatibility between the Windows and Linux platforms.

Contribution: The SamSec security pipeline is a single, unified, and comprehensive solution that integrates several of the best open-source security applications into one consolidated pipeline. This Paper describes the Structure of the complete security pipeline, describes the engineering challenges of using Linux (or other Linux security tools) in conjunction with Windows using WSL (Windows Subsystem for Linux), and presents a complete comparison with traditional or legacy scanning techniques.

II. BACKGROUND AND LITERATURE REVIEW

Through a review of previous research, there are four broad categories of Automated Scanner Operation: Subdomain Enumeration, Port Scanning, Content Discovery and Vulnerability Analysis or Assessment.

A. SUBDOMAIN RECON

1. Active Subdomain Recon Tools (e.g., Amass and Gobuster): These tools usually performs DNS bruteforce attack to find the active subdomains of the Target. This method is more complete than Passive Enumeration; however, it creates much more network traffic as it requires a lot of server- to-server interaction, thus leaving a large footprint on the target's logs and a good chance of being detected via firewalls. The Amass Project hosted on OWASP is the main reference used for Advanced Enumeration techniques and is heavily referenced in academic literature, but it does have the following flaws; high memory usage and long execution times, thus making it ill-suited for smaller, faster automation. [5].

2. Tools with Passive Enumeration Capabilities (Subfinder): Tools with a passive enumeration capability query third-party dataset without sending any packets to the target (an example of this type of tool is Subfinder, created by ProjectDiscovery). The Subfinder tool has a modular API structure that allows it to query from over 30 different sources. Studies/observations has shown from recent testing Subfinder happens to be the best combination of speed and coverage in the early phase of reconnaissance. SamSec uses this method of passive enumeration to reduce time spent scanning for hosts, and the risk of IP being blocked.

B. PORT SCANNING

Port scans verify availability of listening services on the host machines that were found. The standard method used for port scanning is Nmap which has been built using a synchronous or semi-synchronous model. Nmap is considered to have the highest level of accuracy, however due to the level of detail Nmap provides in port scanning, scanning an IP range can take a considerable amount of time.

1. SYN & TCP Connect Scanning: Traditionally, scanning was done in an entire TCP Connect methodology that performed a three way handshake by utilizing a SYN, SYN-ACK, and ACK packet to establish a connection. This process often created a significant amount of traffic and noise across a network, performing scans at a much lower speed than what they are capable of taking advantage of today through their methods of Stealth scanning or SYN scanning. Stealth scanning sends a SYN packet to the target's open port and waits for the SYN-ACK reply, however, the ACK response is not needed to complete the TCP connection. This is accomplished by the modern scanning equipment allowing the use of a RST packet, to reset the connection from the target host instead of waiting for the ACK response, thus improving scan performance.

2. Naabu (ProjectDiscovery): Naabu is a fast-modern port scanner implemented using the Go programming language and using pcap (packet capture) libraries to perform asynchronous SYN scans with the ability to send thousands of requests to any given port each second without having to wait for the full connection timeout. According to the

literature, Naabu is superior to other scanners for use in automation pipelines because it can produce outputs in the native JSON format and can read input directly from many other tools.

C. VULNERABILITY ASSESSMENT

The pipeline's function is to identify any weaknesses in the security of the systems being analyzed.

1. Nikto & Nessus as Legacy Scanners: There are several different types of tools available for checking for vulnerabilities on your systems. For example, Nikto is one of the many tools available to scan for vulnerabilities as it only checks for vulnerable servers using a static database of known vulnerabilities. However, while it can be used on older versions of server systems, organizations are looking for more Options and Nikto provides limited options available; therefore, this tool may not be the best option for all companies. Other tools that provide more coverage for organizations are Nessus tools; however, since they are a commercial product, they may not be usable for student led projects or other lightweight research items.

2. Nuclei is a new way to scan the internet using a "community-powered" method of scanning. Regular tools used to scan for malware or security holes rely on coding rules and are therefore unable to write their own code or rules based on current events and disclosures. Instead, this tool allows users to write their own code/script and rules in a simple, easy-to-learn language (YAML). With this approach, anyone around the world has the ability to write and release templates to check for new vulnerabilities and exposures within hours after they have been discovered. Nuclei has many advantages over traditional scanning tools, including greater flexibility in how the software is used and fewer false positives when scanning.

D. RECENT IMPROEMENTS IN AI SCANNING

The advancements in AI-enabled scanning in the literature published in 2023 to 2025

1. The tools that are being developed now to scan for security vulnerabilities in web applications are based on the use of "Large Language Models" (LLM) which enable application security scanning, which was presented by Stafeev et al. as "YuraScanner" at the NDSS symposium in 2025 [3] In addition to the use of LLMs, a large amount of graphics processing unit (GPU) processing power will be required for the processing of the LLM data.

2. Distributed Vulnerability Management (DVM) is a new way of managing vulnerabilities in the modern world that is scalable and a way to view vulnerability management in a big picture [2].

3. In 2023 CE, there was the need for integration of tools to address "Data Silos" [1] when using DVM, and it is becoming increasingly necessary to have a consolidated and unified framework to prevent the fragmentation of security tools.

III. SYSTEM ARCHITECTURE (THE SAMSEC PIPELINE)

The first step in the pipeline of SamSec (a suite of micro-services) involves collecting information about existing subdomains of a domain name.

E. STAGE 1: RECONNAISSANCE (SUBFINDER)

We have used Targets (url / domain name / ip address) as the Input (for example, 'example.com').

- After that it uses the output of Subfinder, Which pipeline involves running the 'subfinder' output list through a Python wrapper (the wrapper is responsible for invoking the binary). Additionally, to improve the reliability of tests performed during the academic testing phases,
- The output of this command is stored in first 30 results also known as Scope. This limits the expansion of a test to a single scan which can produce countless subdomains.

The end result of this step is a list of FQDNs that have been deduplicated.

F. STAGE 2: HTTPX

The input into this step is the results from the first stage.

- Here we have use 'HTTPX' to probe hosts that may be open to HTTP or HTTPS connections. This probe will follow redirects issued by a web browser and collect metadata on a HTTP or HTTPS host.
- SamSec optimizes the configuration of HTTPX to filter out any host that returns a 404 (not found) or 5xx (server error) response so that any subsequent scans (which require more resources) are only performed on hosts that are valid, reachable, and accessible.

G. STAGE 3: PORT SCANNING (NAABU)

- The process of validating hosts includes scanning for open TCP ports using the Naabu tool. The scanner is configured to only scan for the top 100

most commonly used ports instead of scanning the whole range of ports (65,535), which speeds up the scanning process.

- Normalization comes from the output generated by Naabu which is in a line formatted JSON Schema and SamSec's parser will consolidate the output from Naabu into a dictionary format with the following syntax {"host": "ip:port"}.

H. STAGE 4: NUCLEI

- The scanning process is the most CPU intensive phase, as Nuclei will conduct a scan using a collection of high-impact exploit templates against any live open TCP Ports and IPv4 Hosts. The types of templates used during this phase may include CVE, Misconfigurations and Default Credentials.
- Severity of the findings will be tagged as Critical, High, Medium, Low and Info on the Threat Severity Scale.
- SamSec utilizes the template-based logic produced by Nuclei to reduce false positive results versus heuristic scanners because typically Nuclei requires multiple matchers (i.e., one specific Status Code AND one specific Regex match).

I. STAGE 5: DNS RESOLUTION (SAMDNS)

Dig and other similar tools output results that are not easily processable programmatically.

- To Complete this task, we have used a Python module (SamDNS) utilizing Dnspython.
- This tool has been designed in such a way that allows us to Collect multiple types of records (including A records, AAAA records, MX records, TXT records, NS records, and CNAME records). SamDNS also returns clean JSON output of the records returned so that you can easily visualize the infrastructure of your target (for example, you can see if the target is hosted behind Cloudflare).

IV. IMPLEMENTATION AND CHALLENGES

While developing This tool we faced numerous challenges specially in the process of combining Linux security tools with a Windows-hosting environment.

A. WSL (WINDOWS SUBSYSTEM FOR LINUX)

There were some infrequent engineering difficulties in the engineering process of integrating Linux security tools and windows hosting platform to make up Samsec. It should be noted that

- A. Cross Platform Orchestration (WSL), the language and its architecture brighten this technology further in the same section of the Paper. A Architecture Isomorphicity (WSL) Most security tools with high performance are specifically coded on a Linux base (like Nuclei, Naabu). When they are executed directly on windows they are subject to issues such as path errors and dependency conflicts. The WSL Bridge: It is a bridge that has been constructed due to SamSec connecting towards a Linux subsystem on Windows (WSL). Some shell command preparation is done by our Python backend (on windows) and shells the Linux binaries using WSL.
- Command Wrapper:

```
def run_wsl_command(command):
wsl_cmd = f"wsl {command}" process
= subprocess.Popen(
wsl_cmd, stdout=subprocess.PIPE,
stderr=subprocess.PIPE
)
return process.communicate()
```

- Path Translation: File processing was a grave problem. Windows opera uses backslashes (C: Users) and Linux uses forward slashes (/ mnt c Users). SamSec implements the idea of dynamic path translator such that files created by Linux applications can be interpreted by the windows Python backend.

B. ASYNCHRONOUS TASK MANAGEMENT

The time taken to scan a target will be between 30 and 30 minutes. However, a normal scanning operation provided by a synchronous HTTP operation would fail. Solution:

- FastApi BackgroundTasks Verified have assisted in the implementation of the scanning process.

Workflow:

- A request is sent to the /api/scan.
- The server will, in turn, give the unique scanid and will reply with 202 Accepted immediately. The scanning stream starts at a background thread.
- Request to the server (i.e. GET /api/status/{scanid}) to modify the UI progressbar will be made via Frontend.

C. Data Normalization Schema

The output of the two instruments is varied. Output of subfinder is plaintext, output of Naabu is JSON-lines and output of Nuclei is compound JSON objects.

- Unified schema: To ensure the results are the same, we designed only one schema in the form of a JSON, such that all of the tools use the same schema:

```
{
                          "scan_id":
                              "uuid"
                                   ,
                        "timestamp":
                            "iso860
  1", "assets":                          {
 "subdomains":                  [...],
 "live_hosts":                   [...]
  },
  "findings":                            [
    {
      "tool":                   "nuclei",
      "severity":                 "high",
      "name":       "SQL       Injection",
      "host":             "api.target.com"
    }
  ]
}
```

The schema helps the frontend of the React to display the result of the different tools in a non-partisan format.

V. COMPARATIVE ANALYSIS

To have a better and deeper understanding of the efficiency of SamSec, we conducted a comparative study against standard industry frameworks.

A. Feature Comparison

TABLE I COMPARATIVE ANALYSIS

Feature	OWASP ZAP	Nmap (Scripting Engine)	Burp Suite Pro	SamSec (Proposed)
Primary Use	DAST / Proxy	Network Discovery	Manual Pentest	Automated Recon
Enumeration	Limited	Active Only	Passive (Limited)	Passive (Multi-Source)
Speed	Slow	Moderate	Slow	High
Automation	API (Complex)	Scriptable	API (Paid)	Native Pipeline
Cost	Free	Free	High (>$450+)	Free / Open Source
Architecture	Java (Heavy)	C/Lua	Java (Heavy)	Python / Go (Light)

B. PERFORMANCE METRICS

In a controlled test against a bug bounty target that is a permissioned network (Scope:*.example.com) SamSec completed its reconnaissance and vulnerability scans in 3 minutes and 12 seconds while OWASP ZAP (Automated) completed the same scope (- (active) crawling) in 14 minutes and 45 seconds.

1. By utilizing passive enumeration (Subfinder) and fast SYN scanning (Naabu) SamSec provided the ability to create a vulnerability scan at approximately 4 times the speed of OWASP ZAP's active crawling.
2. Resource Utilization of Scanners: During times of peak scan activity SamSec used on average 450MB of RAM (due to the use of Go which has an effective memory management).
3. Java based scanners such as Burp Suite have consistently utilized more than 2GB of RAM during peak activity.

VI. LIMITATIONS

Although SamSec allows for greater speed and automation of vulnerability scanning it also has some limitations:

1. Depth vs Speed: As SamSec relies on passive enumeration and template-based scanning, it is possible that it may miss some "logic bug" vulnerabilities (Insecure Direct Object References for example) that may require a more in-depth manual assessment of the application.

2. Authentication: The current project is mostly tailored for unauthenticated external scanning and does not currently allow for complex login flows and session management.

3. WSL (Windows Subsystem for Linux) Dependency: The requirement to use WSL may result in the dependency of WSL being limited to the configured host OS and will limit the overall portability to Linux only server environments without the use of Docker Containerization

VII. FUTURE SCOPE

The modularity of SamSec provides numerous

possibilities for growth into the future:

1. Cloud-Native Architecture - Changing from utilising a local Windows Subsystem (WSL) instance for running SamSec, to running the application through a Docker/Kubernetes based architecture would allow for decentralizing the scanning process; breaking out the reconnaissance tasks across numerous machines.
2. Visual Reconnaissance (visual evidence or Support) - Using applications/ functions, such as gowitness, that are capable of automatically takeing screenshots of current websites(targets) to visually identify phishing login pages and potentially unsecured dashboards.
3. Reporting - The ability to export the JSON scan results as PDF compliance reports in accordance with ISO 27001/OWASP Top 10.

VIII.CONCLUSION

The increase of the digital attack surface at such a rapid pace means that organisations will need to shift away from the manual reconnaissance approach of traditional penetration testing and implement an automated, pipeline-driven security assessment process. SamSec is a new approach to integrating current, modern open source tools into a unified scanning framework. Using the speed of Go based binaries Subfinder, Naabu and Nuclei in combination with the orchestration capabilities of Python, we can provide a fast, lightweight and reliable means of performing the first step of vulnerability assessment.

with our benchmarks showing that SamSec is considerably faster and uses significantly fewer resources than traditional tools; therefore SamSec is optimal for academia and embedding Continuous Integration(Pipeline also known as CI/P). While it does not replace the need for experience and skill required to perform manual penetration testing, it serves as a very effective force multiplier for automating the repetitive "low hanging fruit" detection and allowing analysts to concentrate on the more complicated and higher impact findings.

REFERENCES

[1] M. Al-Hawamleh, M. Hammad, and A. Al-Sadi, "Automated Web Vulnerability Scanner Framework," *International Journal of Computer Techniques*, vol. 10, no. 2, 2023.

[2] M. Albahar, " Distributed Vulnerability Management System," *Journal of Cybersecurity and Privacy*, vol. 5, no. 1, pp. 45-58, 2025.

[3] M. Albahar, "Distributed Vulnerability Management System," *Journal of Cybersecurity and Privacy*, vol. 5, no. 1, pp. 45-58, 2025.

[4] S. Stafeev, T. Recktenwald, et al., "YuraScanner: Leveraging LLMs for Task-driven Web App Scanning," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, 2025.

[5] ProjectDiscovery, "Nuclei: Template-based Vulnerability Scanner," *GitHub Repository*, 2024.

[6] OWASP, "Amass Project Documentation," 2024. G. Lyon, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Insecure.com LLC, 2009.

[7] ProjectDiscovery, "Naabu: A fast port scanner written in Go," 2024.

[8] ProjectDiscovery, "Subfinder: Passive subdomain discovery tool," 2024.

[9] R. S. Sarpong and J. Larbi, "Performance Evaluation of Open Source Web Application Vulnerability Scanners based on OWASP Benchmark," *International Journal of Computer Applications*, 2022.

[10] S. Balwante et al., "Design and Implementation of an Enhanced Web Application Vulnerability Scanner," *International Journal of Innovative Research in Computer Science and Technology*, vol. 13, 2025