ISSN: 2581-3404 (Online) IF: 5.86 (SJIF)
IJIRTM, Volume-10, Issue-3, June-2026

SamSec: A Distributed Orchestration Framework for Unified Attack Surface Management and Dynamic Security Testing

[1]Vaishnavi, [2]Janvi Tyagi, [3]Dr. Shakti Arora, [4]Mrs. Shruti Jindal
[1,2]Student, [3]Head & Professor, [4]Assitant Professor
[1,2,3,4] Department of Computer Science & Engineering (Cyber Security)
[1,2,3,4] Panipat Institute of Engineering and Technology, Samalkha, Panipat, Haryana, India.
[1]vaigau2105@gmail.com, [2]tyagijanvi257@gmail.com, [3]hod.cyber@piet.co.in, [4]shruti.cse-et@piet.co.in

Abstract. *Many organisations with large external asset inventories have one challenge in front of them: there are brilliant sets of sophisticated tools available, but the lack of an efficient combination of these tools remains an issue. Manual data format conversion, tool-to-tool hand-offs, and brittle scripting all contribute to what we call the "Orchestration Gap"—in the amount of engineering hours that are spent per scan cycle data entry instead of data analysis. This paper introduces the distributed orchestration framework SamSec that overcomes this gap by utilizing a 6-step reconnaissance funnel based on passively discovered subdomains, DNS health analysis, high-speed port scanning, endpoint probing, web crawling, and template-based vulnerability checks. SamSec executes tools in an asynchronously distributed manner via Celery, Redis [10] for this, it also provides normalization of all subsequently executed tools, enabling all of such outputs to be presented in a common JSON schema, it also integrates the MITRE ATT&CK adversarial tactic engine [12] to map the vulnerabilities that are discovered during the process to the adversary's tactics. The progressive funnel architecture brings up to 20X less redundant compute than naïve approaches. SamSec allows you to assess your external attack surface continuously and comprehensively—in minutes—in the same short time frame that it takes engineers to assess external attack surface while avoiding the need to engage in format-conversion tasks. Technology—Attack Surface Management, Tool Orchestration, Automated Vulnerability Assessment, Distributed Systems, DevSecOps, and Asynchronous Processing are example technologies. Technologies—Example technologies include Attack Surface Management, Tool Orchestration, Automated Vulnerability Assessment, Distributed Systems, and DevSecOps, among others, plus Asynchronous Processing.*

Keywords: Attack Surface Management, Tool Orchestration, Automated Vulnerability Assessment, Distributed System.

Introduction

A. The Orchestration Gap

Everyone who runs their security team in the 100+ asset range is experiencing frustrations with reconnaissance workflow. In a best-of-breed scenario, there are tools like Subfinder [1], Naabu [2],

HTTPX [3] and Nuclei [4] that have one specific focus and excel at it, but getting them all to work together can take precious hours of engineering time per cycle of the scan.
The process used in practice is as follows: The practitioner executes Subfinder [1], saves the plain text output, manually removes any duplicates, copies the list to Naabu [2] for port scanning, processes the JSON-lines output using the custom script developed by the practitioner, feeds the list of live hosts to HTTPX [3] and the filtered list of active URLs to Nuclei [4]. This generally will take 45-60 minutes for the entire scan. Companies spend money on best-of-breed security products, and then engage their top engineers as data-entry guys to work with the products to make them work and convert the data in a compatible format.
A cascading failure is a situation that occurs repeatedly, causing three failures in the Orchestration Gap.
Inconsistency: Manual methods generate uneven subdomain and port coverage and results are difficult to compare between practitioners and sessions.
Latency: If one scan cycle takes one hour of engineering effort, it is impractical to repeat a scan every weekly or continuously. Orchestration uses up analysis/recovery time.
We have Data Integration Friction: Subfinder [1] emits plaintext; Naabu [2] emits JSON-lines; Nuclei [4] emits nested JSON; HTTPX [3] emits JSON arrays. These tools need to be connected via a fragile and hard to maintain custom parsing logic.

B. SamSec's Contributions
SamSec, built from the ground up to eradicate the Orchestration Gap, is a purpose built orchestration platform. It has 5 main contributions:
State-Aware Automated Pipeline: An automatic pipeline with six pipeline stages, with the results of each stage being consumed by the next one in an automatic fashion. No manual intervention, no format conversion and no orchestration overhead.
SamSec processes multiple concurrent scans without blocking with built-in support for asynchronous execution, using Celery and Redis [10] libraries. Workers send the entire pipeline asynchronously: scan requests immediately return with a scan ID.
All tool outputs are stored in JSON format, which enforces a common schema (Universal Data Normalization), for tool-agnostic downstream processing, correlation and analysis.
Adversary-Centric Threat Intelligence: Vulnerability items discovered are associated with adversary-centric MITRE ATT&CK techniques, rather than to arbitrary CVSS scores.
Any domain resolved through the same system will be progressively examined until 100 candidate endpoints are revealed, at which point the expensive Vulnerability Scanning will be executed on only those 100 endpoints. (Infinitely better than a recursive approach that examines each domain in full to the last endpoint).

C. Paper Structure
In section II, the authors explore the related work. In Section III, a description of system architecture and the six-stage pipeline is provided. Section IV deals with implementation. MITRE ATT&CK integration is covered in Section V. Evaluation of Section VI is in this booklet. Sections VII and VIII discuss limitations and future work.

Related Work

A. Zero Trust Security

With the quick expansion of infrastructure, static annual penetration testing isn't enough. The internet-wide asset discovery pioneers are Shodan and Censys and are search engines, not an assessment platform. Alternate solutions such as commercial ASM solutions (Palo Alto Cortex Xpanse, Rapid7 Attacker View, Censys ASM) are closed source, expensive, and do not integrate with existing toolchains for monitoring. Open source solutions only solve part of the issue. The project OWASP Amass [5] is excellent passive subdomain enumeration from 30+ sources, however is missing the vulnerability assessment feature. Subfinder [1] is lightweight and fast, but you have further it needs to go. Current ASM tools solve about 10% of the reconnaissance problem: subdomain discovery means only that we're seeing visibility, and the other pieces—port scanning, endpoint probing, web crawling and vulnerability auditing—are being ignored.

B. Network Scanning and DAST

There is the canonical network reconnaissance tool: nmap [6] which is widely capable, extensible, features-packed, but it's written in Lua! Its synchronous design does, however, necessitate waiting for each host for full TCP/30 second timeouts, which causes serious performance degradation problems at scale: a scan of 1000 hosts spanning a 1000 ports can take hours in a poor network environment. Naabu [2] does reduce that but still not by 10 times, instead it sends a million SYN packets per second with a tool offering a speed boost of ~10 times by introducing asynchronous SYN scanning. Manual Scoping: manual platforms, such as OWASP ZAP [7] and Burp Suite Professional [14] require manual participation and are optimized for time-boxed assessments and for human operators; are not good at large scale; and produce heuristic detections that result in false positives. Nuclei is designed to overcome these drawbacks with the help of templates: AND-logic multi-matcher templates (logical scanning operators that traverse all the content of living-free by the YAML), community-driven template coverage, and quick deploying of templates for newly published vulnerabilities. In SamSec, these approaches are all combined in an orchestration.

The practical development of orchestration and distributed systems.

Practitioners deal with tool orchestration in a variety of ways, either with ad-hoc shell scripts (causing fragile code, which collects technical debt), custom Python automation (which requires basic XML expertise, yet every organization is trying to build it themselves), or with commercial products such as Swimlane and commercialized Rapid7 InsightConnect (which could be prohibitively expensive for enterprise, and are not generally an ideal solution for tool integration use cases). There are numerous point solutions in the security marketplace, but no one has built a protection purpose-built orchestration platform, that is SamSec's value prop. Celery is a proven distributed task queue for Python with support for running long-running tasks asynchronously, with the aid of Redis [10] to keep all tasks from being related to each other. FastAPI [11] provides a way to build the API layer with high concurrency and support for async/await. The microservices' pattern allows independent components scaling and technology heterogeneity [8]. SamSec translates all these into the actual security tool orchestration.

System Architecture

A. High-Level Design

There are 4 Separation Layers in SamSec:

• Presentation Layer: The interface is React.js and is fully responsive throughout long running scans since they don't directly run costly operations.

Orchestration Layer: FastAPI [11] backend that validates incoming requests and enqueues tasks via Celery to various endpoint-specific applications and services, and also gives real-time status of the endpoints. Does not use unnecessary interventions directly (expensive procedures).

Execution Layer: Stateless workers that retrieve tasks from Redis [10] and execute them (using wrappers), normalize the results and save them. Horizontal scaling is easily achieved with Stateless due to a possibility to add workers.

Uses Redis [10] as a task broker, and maintains state using PostgreSQL/SQLite. All state is not to be found in workers.

B. State-Aware Pipelines

The normalization output of each stage is directly fed to the next stage so you don't need to have to intervene between stages. Most important innovation is the consistency of the schema – the records of Stage N exist in a consistent format (despite the developer of the data). This means that there is no need for humans to intervene or have to contort data formats: both are handled by unified schema and that the inter-stage latency is eliminated with direct tool-to-tool data flow.

C. the six-stage Reconnaissance Pipeline.C. the six-stage Reconnaissance Pipeline.

Stage 1 — Passive Subdomain Discovery (Subfinder [1]): Subfinder [1] sends queries to 30+ 3rd-party databases, such as AlienVault, SecurityTrails, Shodan, Censys, BinaryEdge, and others and does not send any packets to the host. Passive DNS enumeration is a type of DNS brute-forcing which does not present any forensic evidence and detection risk. By default, there's a maximum of 50 subdomains (based upon 30 to 100 subdomains). The normalization of unified schema begins prior to Stage 2 (plain text output – 1 subdomain per line).

Stage 2 – DNS Health Analysis (SamDNS): SamDNS is a custom Python application to gather A, AAAA, MX, SPF/DKIM, NS and CNAME info from stage 1's output for each subdomain. It identifies dangling CNAME records (pointers to nonexistent domain) and enables DNS takeover attacks, and allows an attacker to tamper with DNS and e-mail for the affected subdomains. Also for infrastructure intelligence SamDNS identifies mail providers (Google Workspace, Microsoft 365). Stage 2 eliminates unreachable subdomains; down the stream greatly reduced.

Stage 3 - High-Speed Port Scanning (Naabu [2]): This is asynchronously scanning targets with SYN as open and RST as closed status. Naabu [2]'s port scanning is approximately 10 times faster than connection based port scanning, as it sends SYN's, and receives back a SYN-ACK (open) or RST (closed). SamSec targets the top 100 commonly-used ports (80, 443, 8080, 8443, 3306, 5432, 22, 23, 25, 110, 143, etc.), balancing thoroughness with execution speed.

Stage 4—Endpoint Probing and Technology Fingerprinting (HTTPX [3]): HTTPX [3] probes each open port with an HTTP/HTTPS request and will retain all live-endpoints that respond with any return codes from the range 200 to 3xx, 401 or 403. 404s (for missing files) and 5xxs (for server errors) are not followed. All End Point are scanned and digitised in HTTPX [3] to give the identification of the Web Server's Software, JavaScript Frameworks, Programming Languages, WAF Versions, TLS Versions and

CMS Platforms (e.g. WordPress, Drupal, Joomla). This fingerprinting selects the type of template to be used in Stage 6: WordPress detected will give preference to WordPress-themed templates.
Stage 5—Web Application Crawling (Katana [13]): Katana [13] is a breadth-first crawler that is able to extract genuinely dynamically-chained API endpoints from HTML links and JavaScript source. To ensure depth and timeout within apps (3-4 levels) and 5-10 minutes per app). All the information (HTTP request/response) we captured during our crawling process is stored as proof of concept evidence for vulnerabilities found in SamSec.
Stage 6—Community Maintained YAML Based template Vulnerability Auditing (Nuclei [4]): Community maintained YAML templates are used to apply scanned discovered endpoints to them. The parameters that will be sent in the HTTP request and matchers on the response (status codes, regex patterns and json paths) are defined by each template. AND-logic multi-matcher conditions reduce false positives to a minimum. Within hours, vulnerabilities get added to templates when they are discovered by community members! It defends against any known vulnerabilities in the CVEs, against exposed admin panels, against default credentials, against misconfigurations of technology, and against protocol vulnerabilities (weak TLS, expired certificates).
D. Funnel Efficiency
The efficiencies of improvement are measurable. With Nuclei [4] scanning 5,000 templates per 1,000 domains:
• Naive approach: 1,000 × 5,000 = 5,000,000 checks.
SamSec funnel: 50, live endpoints – filtered at DNS – 500 hosts resolving – filtered at port scan on endpoints or endpoint probing – 50 x 5,000 outputs. The number assets that was found and was left with the 90% reduced by 90%.
E. Unified Data Normalization Schema
All Source Outputs: All tools output are adjusted to::
{ "scan_id": "uuid",
"timestamp": "2024-05-14T10:30:00Z",
"stage": 1, "entity_type": "subdomain",
"host": "api.example.com",
"ip": "192.0.2.1", "port": 443,
"protocol": "https",
"source_tool": "subfinder",
"severity": "info", "confidence": 0.95,
"raw_request": "GET / HTTP/1.1...",
"parsed_data": { "status_code": 200,
"response_time_ms": 245 } }
The match against the entity_type: subdomain results are from Subfinder [1] results, the results of Naabu [2] are matched with entity_type: open_port, and finally with the vulnerabilities obtained with Nuclei [4] results, entity_type: vulnerability is found. Advantages: consistency of the rendered front and flexible downstream processing using any tool, linking between results in different tools, consistency of API responses, and historical analysis along the whole longitudinal curve.

Implementation Details

A. Asynchronous Execution

Reconnaissance against actual targets takes 20-40 minutes. The TCP/IP connection to the web server times out after 30–60 seconds. With a synchronous architecture, however, the time elapsed before the completion of a helpful scan would time out. SamSec fixes this: It validates the various parameters submitted in the scan, places an entry to a ScanJob record, submits a Celery task, and immediately returns a response HTTP 202 (Accepted) with a scan ID. HTTP connection is closed. Workers are used to fetch the task from Redis [10] and run the entire pipeline asynchronously. Every 2 seconds a client polls the GET endpoint to check for progress updates.

Operational benefits: Never timeouts while operating on an HTTP based system; supports multiple concurrent users while a scan is running (all activities queued to the set of workers job instances available to them); non-blocking nature (a running scan doesn't affect concurrent users); fault tolerant in case a worker dies (redis [10] stores the state of all pipelines and each one can be taken up by another worker); real time progress visibility as each pipeline stage is completed.

B. Backend with fastapi and API design

Two primary end points: POST /api/v1/scan accepts target_domain, max_subdomains and phase selections, validates with Pydantic; creates an entity called ScanJob, enqueues a Celery task, and returns an entity called ScanJob with state "queued", and an HTTP status of 202. GET /api/v1/scan/{scan_id}, the endpoint returns real-time status (queued/in_progress/completed/failed), status progress (0 – 100), active phase name and partial results to display progress on the frontend.

This is an example of how to write error handlers and Workers.

For each pipeline stage, workers perform the following steps, in sequence: obtain the job from Redis [10] and set the status to in_progress; run each of the tool wrappers; normalize output to a unified schema; persist output; update progress metadata (current_phase, percentage) in the database; aggregate a comprehensive final Report; mark the job as completed. Transient failures are addressed with Exponential backoff retry: Once the third failure occurs a task is permanently marked as failed with diagnostic information stored.

C. The abstraction of Tool Wrapper in D.The abstraction of D Tool Wrapper.

There is one dedicated wrapper for each security tool that will deal with the construction of commands, executed subprocesses with timeout, subsequent parsing of the output, deduplication of such output, placing a cap on the number of results that is returned, as well as standardising error conditions where a tool is not installed, where the subprocess times out, where it does not have permission to execute, and malformed output. The Subfinder [1] wrapper converts subfinder -d {domain} to a subdomain list, has a 5 minute time limit, parses plaintext output, removes duplicates, limits to 50 subdomains to include in the list and outputs a subdomain list. The other wrappers (Naabu [2], HTTPX [3], Nuclei [4], Katana [13], SamDNS) share the same interface patterns, allowing to independently test, and clean the replacement of the tool.

D.WSL Integration and Data Normalization

Brilliantly, Windows users are supported by a seamless bridging of the WSL interface: tool commands are prefixed with wsl, any paths from Windows are auto translated to WSL mount paths (C:\artifacts\scan.json is backslash translated to /mnt/c/artifacts/scan.json), and installation scripts will provision all tool binaries to the WSL space. Once initial setup this bridge cannot be seen by practitioners.

The native output formats of each of these tools get converted to the proper entity type: Subfinder [1] plaintext -> entity_type=subdomain, Naabu [2] JSON-lines -> entity_type=open_port, Nuclei [4] nested JSON -> entity_type=vulnerability, HTTPX [3] JSON array -> entity_type=endpoint, SamDNS record-grouped JSON -> entity_type=dns_record, Katana [13] crawl JSON -> entity_type=discovered_path. Source tool, confidence score and severity metadata is added by each normalizer.
Describe how easily this could be incorporated as a part of

MITRE Attack Framework

Representatives of each individual were present at the workshop.At the workshop, there were representatives from each individual.
The conventional vulnerability reports address the question, "What are the vulnerabilities?The principle vulnerability reports respond to the question "What vulnerabilities exist? Adversaries's WILL be how they would exploit SamSec. Instead, SamSec reports: "Adversary technique T1190 (Exploit Public-Facing Application) leads to initial access; T1098 (Valid Accounts) aids for persistence. This turns a vulnerability list into an adversary capability assessment, which allows leadership to prioritize attacks that have the maximum effect on the adversary: "Our largest exposure is to initial access attacks, so should fix SQL injection and weak credentials first.
B. Mapping Algorithm in mapper.py
Each vulnerability is scored for each of the MITRE ATT&CK techniques and the different factors were weighted and include:
• Vulnerability Class (w_1=0.4): SQL injection → T1190/T1005; RCE → T1059/T1203; default credentials → T1078. Vulnerability class determines technique families, based on dominant factor.
Severity Level (w_2=0.3): CVSS scores were scaled to 0–1. As severity increases such as from a hot fire to a non-fire incident, the amount of impact by adversaries increases; therefore the confidence level when mapping also increases.
Internet-facing root domains are given a score of 1.0, first level subdomains get a score of 0.8, deeper assets get a score of 0.6. Exposing to the outside expands adversary population.
The technology context (w_4=0.1): A compromised PostgreSQL instance has a greater organizational impact than an XSS in a documentation page. The default credentials to SSH are more sensitive than default credentials to services that do not depend critically on the SSH protocol.
Composite score: Score = w_1·Class + w_2·Severity + w_3·Scope + w_4·Technology. All weights add to 1.0, and can be user adjustable.
C. Representative Mappings
SQL Injection: T1190 Exploit Public-Facing Application (0.95); T1005 Data from Local System (0.85).
• Default Credentials → T1078 Valid Accounts (0.95), T1110 Brute Force (0.80).
Compromise Infrastructure (0.90), Man-in-the-Middle (0.70) → DNS Takeover / Dangling CNAME (PAT ター) to.
• Weak TLS/SSL → T1040 Network Sniffing (0.70), T1557 Man-in-the-Middle (0.80).
Exploit Public-Facing Application (T1190, 0.85), Cloud Service Discovery (T1526, 0.75) → authenticated APIs.

Information Disclosure: T1592 Gather Victim Host Information (0.60), T1589 Gather Victim Organization Information (0.65).
The Threat Heatmap is the
D. Coverage Layer.
The Coverage Layer is created by aggregating technique scores from all findings to make a tactic-level Coverage Layer with mapper.py. Example—5 SQL injections, 3 default credential Exposures, Weak TLS:
– Initial Access: SQL injection, Rapid 7 uses Score 0.85 (T1190) / unauthenticated APIs Score T.
• Collection: Exposure Score 0.75 – T1005: Data Exfiltrated in the database via SQL injection.
Persistence: Exposure Score 0.65 (T1078 valid accounts via default credentials).
Use of credentials: Exposure Score 0.49 (T1110 brute force facilitation).
The output contains a structured JSON report and an ATT&CK Navigator layer file (mitre_layer.json) that can be directly loaded into the MITRE ATT&CK Navigator for visualization in a heatmap. Security leadership can then choose remediation to the more obvious tactics and not go through one-by-one through a list of vulnerabilities.

Limitations

A. Speed vs. Depth Trade-off: This occurs when a person's speed of a task would decrease with increasing depth, and depth would increase with speed.
The area of SamSec has good breadth (identification of external attack surface) and lacks application analysis in depth with semantics. Scans are deterministic and identify CVEs, default credentials, issues for certain technologies and misconfigurations. This can't detect vulnerabilities in Application logic such as Insecure Direct Object Reference (IDOR), Business Logic flaws, or complex Authorization Bypass, Race Conditions as it has no insight into what the application is trying to do - only application Tooling can find it this way. SamSec is NOT replacing manual penetration testing (MPT), but it will identify these "low hanging fruit" which enable security engineers to focus in on these "low hanging fruit" which can be attacked by logic.
A. Authentication Limitations
Just temporarily SamSec is a so-called unsingularized external Recon. To scan the credential you would need to use all of the following: Secure credential storage (rotation, encryption); Session state management (all through the pipeline); Multi factor authentication support (TOTP, push etc., security keys); Account lock out prevention (rotation); and Session timeout handling (auto-re-authenticate). It is the most affordable close-term extension, and is capable of locating inner applications and admin panels vulnerabilities.
B. Environment Dependencies and Static Mapping
WSL is the prerequisite for Windows deployment, and restricts the ability to create raw packets and manipulate network interfaces, two things normally used for reconnaissance but not supported by Windows. Correction for WSL whereas the mitigation is planned to be Docker containerization which allows deployment on all platforms.
The number of TTPs mitigated on an ICDP that has Internet access is the same as the number of TTPs mitigated on an ICDP that does not have Internet access, despite the difference in risk profile between these two types of ICDPs. It will be leveraged to enable future LLM to be integrated within the mapping to make it a contextual, dynamic mapping with regards to the organisation asset exposure and context.

Future Work

A. Cloud-Native Architecture

Docker containerization will be used for automated worker scaling, high availability & integrated fault tolerance, and will remove WSL, enabling the automatic discovery of assets from cloud providers inventories from API calls for cloud provider (AWS, Azure, GCP).

Use Oracle XML Gateway and Oracle Identity Management Service to secure your Oracle databases.This course covers securing your Oracle databases with Oracle XML Gateway & Oracle Identity Management Service (IMS).

Integration of LLM will allow to generate automatic remediation guidance according to a context – "This SQL injection is correctable with parameterized queries"; Generation of vulnerability report with organization context and prioritized remediation roadmaps; Generation of an automatic template for If the session state is properly managed by the user and handling the session in between the pipeline stages, then the vulnerabilities in the secure app regions (admin panels, the internal application, the application that is supposed to be secure) can be discovered.

A. – Visual Evidence & Compliance Reporting.There must be a function for the device.

Visual evidence capturing of images (phishing pages, compromised dashboards etc) will be possible with GoWitness integration, using screen shots. The Compliance Reporting Extensions will help harmonize SamSec findings to standards and align it with standards to ISO 27001, OWASP Top 10, PCI DSS, HIPAA and others, handing over report on audit-ready data to regulated entities.

Conclusion

They keep great security tools, but are not able to effectively merge them all. The Orchestration Gap – i.e., inputting and changing data types, manually triggering tools and tasks – is a well-documented engineering failure whose time spent is measurable, resulting in delays and latency, and does impact data analysis struggles because of manually triggered tasks.

This is where SamSec fits in and steps right between; in consolidating open-source tools for the tools that are used in the industry. With Universal JSON schemas, there's no more need to worry about format conversion. For distributed execution, Celery and Redis [10] are used, for this reason, concurrent and non-blocking scans are supported without limits of timeout of HTTP response. The progressive funnel delivers up to 20× load in comparison to a regular Nuclei [4] setup while offering the complete asset coverage with no compromise. The MITRE ATT&CK mapping engine is designed to do just that; it has the capability of mapping a list of vulnerabilities onto adversarial intelligence that matches the real-world adversarial patterns.

There is definitely enormous improvement in performance as evidenced by the dramatic improvement over synchronous DAST tools, reduction in resource usage and linear scalability. Most important of all, SamSec provides a paradigm change to security teams by allowing them to use a 'real time' security lens to the entire organisation's attack surface – rather than an annual one - where it can happen. Actually, the more you're using cloud, the more you're adding microservices, the more you're adding attack surface there is – and building an automated ASM that never sleeps and that's about as luxuriously as it gets these days. Specially designed for enterprise-standard, external attack surface management, SamSec is a proof that an all-best-of-breed, open-source orchestration platform is capable to deliver it to organizations of any size.

References

[1] Project Discovery, "Sub finder: Passive subdomain discovery tool," GitHub. https://github.com/projectdiscovery/sub finder (accessed May 2024).
[2] ProjectDiscovery, "Naabu: A fast port scanner written in Go," GitHub. https://github.com/projectdiscovery/naabu (accessed May 2024).
[3] ProjectDiscovery, "HTTPX: HTTP client with multiple features," GitHub. https://github.com/projectdiscovery/httpx (accessed May 2024).
[4] ProjectDiscovery, "Nuclei: Template-based vulnerability scanner," GitHub. https://github.com/projectdiscovery/nuclei (accessed May 2024).
[5] OWASP, "Amass project documentation," GitHub. https://github.com/owasp-amass/amass (accessed May 2024).
[6] G. Lyon, Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning. Insecure.com LLC, 2009.
[7] OWASP, "ZAP: Automated security testing framework," https://www.zaproxy.org (accessed May 2024).
[8] M. Fowler and J. Lewis, "Micro services," martinfowler.com . https://martinfowler.com/articles/microservices.html (accessed May 2024).
[9] Celery Project, "Distributed task queue," https://docs.celeryproject.io (accessed May 2024).
[10] Redis, "In-memory data structure store," https://redis.io (accessed May 2024).
[11] S. Ramírez, "FastAPI: Modern web framework," https://fastapi.tiangolo.com (accessed May 2024).
[12] MITRE Corporation, "ATT&CK Framework," https://attack.mitre.org (accessed May 2024).
[13] ProjectDiscovery, "Katana: Web crawling and reconnaissance engine," GitHub. https://github.com/projectdiscovery/katana (accessed May 2024).
[14] PortSwigger, "Burp Suite," https://portswigger.net/burp (accessed May 2024).
[15] Docker, "Container platform," https://www.docker.com (accessed May 2024).