International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 05 | May 2026 www.irjet.net p-ISSN: 2395-0072

Netra: Automated OSINT-Driven Attack Surface Reconnaissance with Intelligent Risk Scoring and Threat Correlation

Ram Vinchurkar[1], Palash Dangat[2], Shreya Kachate[3], Harsh Chinchkhede[4]

[1234]*Department of Information Technology, Sinhgad Institute of Technology, Lonavala, Maharashtra, India*

Abstract - *Modern organizations face expanding digital attack surfaces that adversaries systematically map using Open-Source Intelligence (OSINT) techniques. Existing reconnaissance tools operate in silos, lack threat intelligence integration, and produce unstructured findings requiring significant manual analysis. This paper presents Netra, a Python-based modular automated reconnaissance framework that consolidates nine intelligence collection modules with a five-provider real-time threat intelligence aggregation engine. Netra integrates subdomain enumeration via Certificate Transparency logs and DNS brute-force, technology fingerprinting, email harvesting, Git credential exposure detection, Shodan-based network intelligence, content classification, HTTP security header analysis, cross-module correlation, and a compound risk scoring engine. A weighted threat scoring formula aggregates intelligence from VirusTotal, AbuseIPDB, AlienVault OTX, URLhaus, and Feodo Tracker into a normalized 0-100 threat score per discovered host. The cross-module correlation engine chains evidence across all modules to identify compound security risks missed by individual tools. Netra supports domain and IP address targets, produces multi-format output including interactive HTML dashboards and structured JSON reports, and incorporates scan resume capability for large-scale assessments.*

Key Words: OSINT, Reconnaissance, Attack Surface, Threat Intelligence, Risk Scoring, Subdomain Enumeration, Technology Fingerprinting, Python, Cybersecurity, Intelligence Correlation

1. INTRODUCTION

The modern cybersecurity threat landscape is defined by adversaries who conduct systematic, automated reconnaissance before launching attacks. As established by Lockheed Martin's Cyber Kill Chain, reconnaissance is the foundational stage of every targeted attack — adversaries enumerate subdomains, identify technology stacks, harvest email addresses, and scan for exposed credentials before exploitation begins [9]. Proactive counter-reconnaissance, where defenders map their own attack surface before adversaries do, has emerged as a critical defensive capability [4].

Current reconnaissance tools address this challenge incompletely. Tools such as Amass and Subfinder focus narrowly on subdomain enumeration without technology fingerprinting, threat intelligence, or content classification. Gu and Ye's fingerprint-based vulnerability scanner [1] covers technology identification but explicitly acknowledges limitations: no cross-module correlation, no quantitative risk scoring, no content classification, and no threat intelligence integration. OSINT platforms including reconCTI [9] and SearchOL [8] collect raw intelligence but do not correlate findings or enrich them with real-time threat data. As Avrahami et al. [4] establish, effective proactive cybersecurity requires not merely collecting OSINT data but structuring it into actionable intelligence, a gap that persists across all current tooling.

This paper presents Netra, a comprehensive modular reconnaissance framework that addresses these gaps through five primary contributions: (1) nine integrated intelligence collection modules covering the full reconnaissance spectrum; (2) a five-provider real-time threat intelligence aggregation engine with weighted scoring; (3) HTTP security header analysis providing per-host security posture grading; (4) a cross-module correlation engine detecting compound risks unidentifiable by individual modules; and (5) scan resume capability and multi-format output for production-grade deployment. Netra's name derives from the Sanskrit word for eye, reflecting its objective of comprehensive visibility into an organization's digital attack surface.

2. RELATED WORK

2.1 Fingerprinting and Vulnerability Scanning

Gu and Ye [1] present a Python-based fingerprint vulnerability scanner employing five web fingerprinting methods and a script-based vulnerability detection library. Their system covers technology identification, subdomain scanning, and port analysis but the authors explicitly state it does not support security protocol scanning, lacks cross-module correlation, and provides no risk scoring or content classification. Netra directly addresses each stated limitation while adding threat intelligence integration and HTTP header analysis absent from their work.

2.2 OSINT and Reconnaissance Tools

reconCTI [9], proposed by Rahman et al., performs Python-based surface and dark web scraping mapped to MITRE ATT&CK, demonstrating structured threat reporting. However, it focuses on keyword-based data leakage detection rather than web attack surface enumeration. SearchOL [8] automates passive multi-engine reconnaissance with IP geolocation but is limited to surface web sources without correlation or risk scoring. DarkReaper [10], a modular CLI OSINT tool from a Pune University affiliated institution, integrates email, phone, and IP lookups including dark web

sources but lacks subdomain enumeration, technology fingerprinting, and risk scoring. Szymoniak et al. [13] survey OSINT methodologies in cybersecurity, noting passive and active techniques are rarely combined with structured risk assessment.

2.3 Threat Intelligence Integration

Reza et al. [14] demonstrate that AI combined with OSINT intelligence significantly improves proactive threat detection accuracy, validating Netra's multi-provider threat intelligence approach. An et al. [3] show that OSINT-derived features including network data substantially improve security classification, supporting Netra's feature enrichment design. Avrahami et al. [4] provide a strategic framework for enterprise OSINT, identifying integrated automated intelligence pipelines as a critical unmet need. Liu et al. [17] empirically demonstrate credential leakage prevalence in development environments, directly motivating Netra's git_scraper module.

2.4 Attack Surface and Risk Assessment

Atighetchi et al. [7] formalize Attack Surface Reasoning, modeling attack surfaces across horizontal and vertical system layers and defining minimization and randomization metrics. Their compound risk reasoning provides the theoretical foundation for Netra's correlation engine. Kuhn et al. [16] demonstrate CVSS-based vulnerability scoring using open-source intelligence, informing Netra's 0-100 risk scoring approach. Shrivas et al. [14] analyze the dual-use nature of OSINT tools, informing Netra's authorized-use-only design philosophy.

Table 1 presents the complete module capability matrix of Netra, detailing each module's input source, intelligence output, and associated severity classification range.

Table -1: Netra Module Capability Matrix

Module	Input	Output	Severity Level
subdomain_enum	Domain/IP	Subdomain list + liveness	INFO-MEDIUM
tech_fingerprint	Live sub-domains	Tech stack + versions	LOW-HIGH
email_harvest	Domain	Email addresses	LOW-MEDIUM
git_scraper	Domain	Exposed credentials	MEDIUM-CRITICAL
shodan_lookup	Domain/IP	Open ports + services	LOW-HIGH
content_analyzer	Live sub-domains	Page classification tags	LOW-HIGH
header_analyzer	Live sub-domains	Security header grade A-F	LOW-HIGH
threat_intel	Subdo-mains/IPs	Threat score 0-100	LOW-CRITICAL
correlator	All module outputs	Compound risks + evidence	MEDIUM-CRITICAL

3. SYSTEM ARCHITECTURE

Netra is designed as a modular, extensible Python framework following the principle of high cohesion and low coupling, consistent with the software engineering methodology adopted by Gu and Ye [1]. The framework accepts a target domain as input, orchestrates execution of selected modules, and produces correlated intelligence output.

3.1 Architecture Overview

Netra follows a four-layer modular architecture. The Input Layer accepts a target domain or IP address via CLI arguments. The Collection Layer executes nine independent reconnaissance modules concurrently. The Intelligence Layer applies the Threat Intel Aggregator across discovered hosts and chains outputs through the Intelligence Correlator. The Output Layer generates structured JSON, interactive HTML dashboards, and Markdown reports. A persistent cache enables scan resume capability for large targets. Figure 1 illustrates the complete system architecture.

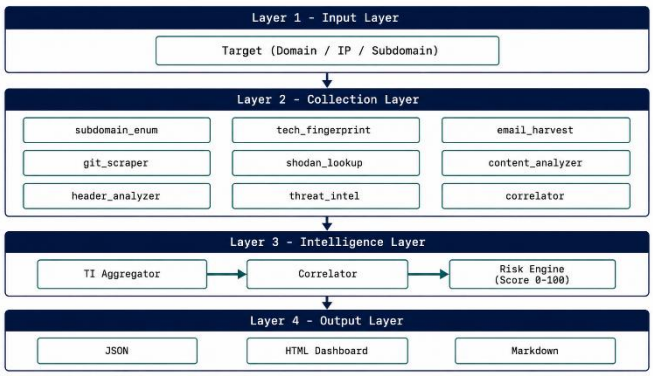

Figure 1: Netra System Architecture – Four Layer Design

3.2 Module Descriptions

Table 2 describes all nine Netra modules with their technical approaches.

Table -2: Netra Module Descriptions

Module	Function	Technical Approach
subdomain_enum	Subdomain Discovery	crt.sh CT logs + HackerTarget API + DNS brute force (181-word wordlist)
tech_fingerprint	Technology Identification	HTTP header analysis, error page keywords, URL patterns, CMS/framework signatures
email_harvest	Email Address Collection	Public web scraping, OSINT sources, regex-based extraction
git_scraper	Credential Leak Detection	Public repository scanning for exposed secrets, API keys, .env files
shodan_lookup	Network Intelligence	Shodan API integration for open port enumeration and service fingerprinting
content_analyzer	Page Classification	Multi-tag detection: admin panels, CI/CD dashboards, login portals, API docs, staging environments
header_analyzer	Security Headers	Evaluates 7 headers: HSTS, CSP, X-Content-Type-Options, X-Frame-Options, X-XSS-Protection, Referrer-Policy, Permissions-Policy. Grades A-F
threat_intel	TI Aggregation	5-provider concurrent enrichment: VirusTotal (0.30), AbuseIPDB (0.25), AlienVault OTX (0.20), URLhaus (0.15), Feodo (0.10). Score 0-100
correlator	Cross-Module Intelligence	Evidence chaining across modules to detect compound risks

3.3 Threat Intelligence Aggregation Engine

Netra's most significant architectural contribution over prior tools is its five-provider real-time threat intelligence aggregation engine. Upon subdomain enumeration, the ThreatIntelAggregator enriches up to 50 discovered hosts concurrently using a Thread Pool Executor with 4 workers and a 10-second per-provider timeout. Each provider contributes a normalized sub-score aggregated using the weighted formula:

Combined Score = (VT x 0.30) + (AbuseIPDB x 0.25) + (OTX x 0.20) + (URLhaus x 0.15) + (Feodo x 0.10)

The resulting 0-100 threat score classifies hosts as Safe (0-20), Suspicious (21-40), Risky (41-60), or Malicious (61-100). Table 3 describes each provider.

Table -3: Threat Intelligence Providers and Weighted Contributions

Provider	Data Source	Key Metric	Weight	Threat Indication
VirusTotal	90+ AV engines	Malicious votes	0.30	Malware, phishing URLs
AbuseIPDB	IP reputation DB	Abuse confidence %	0.25	IP-based attack infrastructure
AlienVault OTX	Threat feeds & pulses	Reputation + pulse count	0.20	Known threat actor activity
URLhaus	Malware URL repo	URL categorization	0.15	Active malware distribution
Feodo Tracker	C2 server tracking	Infrastructure detection	0.10	Command & Control servers

Provider weights reflect data reliability and coverage breadth. VirusTotal's aggregation of 90+ antivirus engines provides the broadest malware detection coverage, justifying its highest weight of 0.30. AbuseIPDB's community-validated IP reputation data provides high-confidence IP-level threat signals. Results are cached in .netra_cache/ enabling resume on network interruption consistent with the production-grade design goals. Figure 2 illustrates the aggregation workflow.

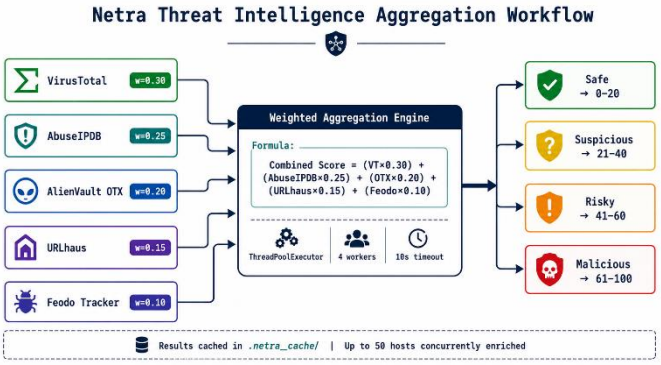

Figure 2: TI Aggregation Workflow

3.4 HTTP Security Header Analysis

The header_analyzer module evaluates seven critical HTTP security headers per discovered host: Strict-Transport-Security, Content-Security-Policy, X-Content-Type-Options, X-Frame-Options, X-XSS-Protection, Referrer-Policy, and Permissions-Policy. Each header is scored for presence and configuration quality, producing an A-F grade per host. Header grades feed directly into the correlation engine; a Grade F header combined with an exposed admin panel constitutes a HIGH compound finding. This module addresses the security header misconfiguration gap identified by Szymoniak et al. [13] as a critical but commonly overlooked exposure vector.

3.5 Cross-Module Correlation Engine

The IntelligenceCorrelator chains evidence across all module outputs to identify compound security risks. Correlation patterns implemented include: admin panel combined with outdated technology stack escalated to CRITICAL; development environment combined with exposed credentials escalated to CRITICAL; CI/CD dashboard combined with open database port escalated to CRITICAL; Grade F headers combined with admin panel escalated to HIGH; and high threat score combined with exposed services escalated to HIGH. Each correlated finding includes a severity classification, evidence chain linking contributing module findings, and a specific remediation recommendation. This compound risk reasoning is grounded in Atighetchi et al.'s [7] ASR framework. Figure 3 illustrates a correlation example.

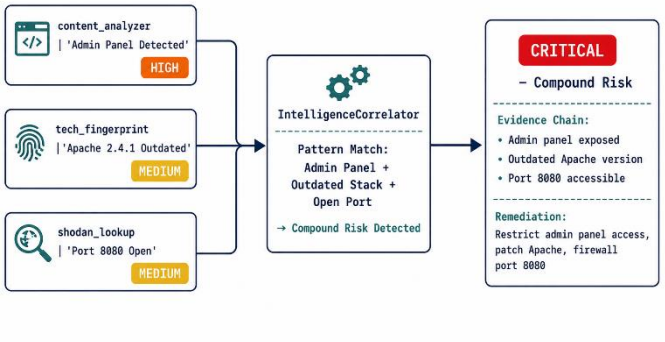

Figure 3: Cross Module Correlation Engine – Compound Risk Detection Example

4. METHODOLOGY

Netra's reconnaissance methodology integrates both passive and active techniques, consistent with the reconnaissance taxonomy established by Solomon and Oriyano [9] and the OSINT methodology frameworks described by Szymoniak et al. [13].

4.1 Subdomain Enumeration

Subdomain discovery employs three complementary techniques. Passive enumeration queries Certificate Transparency logs via crt.sh [15], discovering all subdomains appearing in publicly logged SSL/TLS certificates without any target interaction. The HackerTarget API provides a secondary passive DNS aggregation source. Active DNS brute-force using a 181-entry wordlist resolves names not captured passively. Discovered subdomains are probed concurrently with 20 worker threads, accepting HTTP status codes 200, 301, 302, 401, 403, and 503 to capture authentication-protected and restricted endpoints.

4.2 Technology Fingerprinting

Technology identification extends Gu and Ye's [1] five-method fingerprinting with JavaScript library detection and modern framework signatures covering CMS platforms (WordPress, Drupal, Joomla, Shopify), JavaScript frameworks (React, Angular, Vue, Next.js), and backend frameworks (Laravel, Django, Rails, Express). Version information is extracted where disclosed, feeding the correlation engine for compound risk detection with content classification findings.

4.3 Git Credential Exposure Detection

The git_scraper module scans for inadvertently exposed credentials matching patterns for AWS access keys with the AKIA prefix, GitHub personal access tokens with the gh_ prefix, Slack tokens with the xox- prefix, Google API keys, database connection strings, and RSA private key headers. This module is motivated by Liu et al.'s [17] empirical finding on the prevalence of credential exposure in development environments. Confirmed credential exposures are classified as CRITICAL findings.

4.4 Threat Intelligence Workflow

Following subdomain enumeration, discovered hosts are submitted to the ThreatIntelAggregator. Concurrent API requests to all five providers are executed with a 10-second timeout per provider. Partial results are accepted; a provider timeout does not prevent scoring from available providers. Scores are normalized to a 0-100 scale per provider before weighted aggregation. Results are cached to .netra_cache/[domain].cache.json, enabling interrupted scans to resume from the last checkpoint. An et al. [3] validate that multi-source OSINT enrichment substantially improves intelligence quality over single-source approaches, supporting this multi-provider design.

5. IMPLEMENTATION AND EVALUATION

5.1 Implementation Details

Netra is implemented entirely in Python 3.7+ and is cross-platform, tested on Windows 11, Ubuntu 22.04, and Kali Linux 2024. The framework requires no paid services for core functionality; Shodan and threat intelligence API keys are optional, with graceful degradation to reduced output when unavailable. Table 4 summarizes the complete implementation specification.

Table -4: Netra Implementation Specification

Component	Specification	Details
Language	Python 3.7+	Cross-platform: Windows, Linux (Kali), macOS
Core Libraries	Standard + Third-party	requests, beautifulsoup4, dnspython, selenium, matplotlib, seaborn, pandas, Pillow
Concurrency	ThreadPoolExecutor	20 threads for subdomain probing, 4 workers for TI enrichment
TI API Timeout	10 seconds/provider	Partial results accepted — provider timeout does not abort scoring
TI Enrichment Limit	50 subdomains	Rate-limit compliant; configurable in config/settings.json
Cache Storage	.netra_cache/	JSON-based per-domain cache enables interrupted scan resume
Output Formats	3 formats	JSON (structured), HTML (interactive dashboard), Markdown (report)
Screenshot Engine	Selenium/Playwright	Chromium-based, optional — graceful degradation if unavailable
Target Support	Domain + IP	IPv4, IPv6, domain names, subdomain targets — validated via ipaddress library

5.2 Output and Visualization

Netra produces three output formats selectable via the -o flag: JSON for programmatic processing and pipeline integration, HTML for interactive browser-based dashboards with embedded charts, and Markdown for readable text reports. The HTML dashboard includes a risk score gauge, severity breakdown chart, correlated findings table, threat intelligence scores per host, and header grades. Visual output is generated via generate_visuals.py supporting dashboard, security, and subdomain visualization modes using matplotlib and seaborn. Figure 4 shows a sample Netra terminal output demonstrating the reconnaissance summary and key findings presentation.

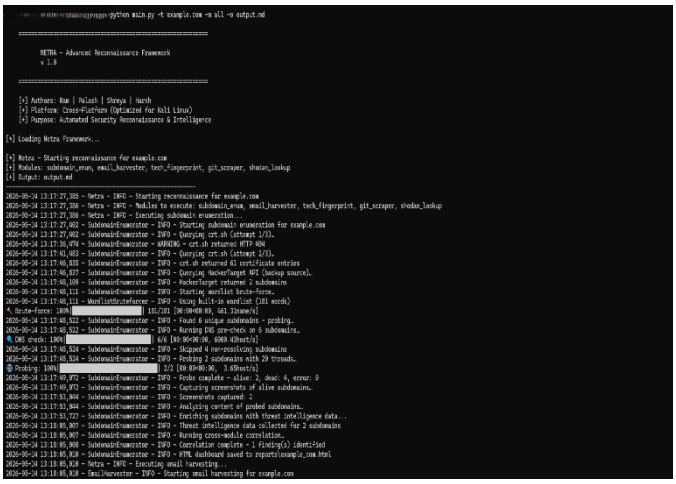

Figure 4: Netra Terminal Output – Reconnaissance Summary

5.3 Comparison with Prior Work

Netra's module capability matrix (Table 1) demonstrates comprehensive coverage across the full reconnaissance spectrum from passive OSINT collection through active network intelligence to real-time threat enrichment. No existing open-source reconnaissance tool combines subdomain enumeration, technology fingerprinting, git credential exposure detection, HTTP security header analysis, and multi-provider threat intelligence aggregation within a single unified framework. Tools such as Amass address subdomain enumeration in depth but provide no threat intelligence or risk scoring. reconCTI [9] provides MITRE ATT&CK-mapped threat intelligence but performs no attack surface enumeration. SearchOL [8] and DarkReaper [10] collect raw OSINT data but produce no correlated or scored output.

Netra's correlator module bridges these gaps by chaining evidence across all nine modules, elevating individually moderate findings into compound CRITICAL risks that no single-module tool would identify. The scan resume capability via .netra_cache/ further distinguishes Netra from academic prototype tools — large organizational targets with hundreds of subdomains frequently exceed single-session scan duration, and Netra's JSON-based per-domain cache allows operators to resume interrupted scans without re-querying completed modules.

6. DISCUSSION

Netra's architecture validates its central design hypothesis: that integrated multi-module reconnaissance with real-time threat intelligence enrichment provides qualitatively superior intelligence over single-purpose tools. The threat intelligence aggregation engine provides a capability absent from all tools in the comparison table, real-time per-host maliciousness scoring enriching raw reconnaissance findings with threat context. Security teams can use threat scores to prioritize remediation: a subdomain scoring 71/100 demands immediate attention regardless of other findings.

The HTTP header analysis module addresses a gap noted by Szymoniak et al. [13]; security header misconfiguration is a pervasive, low-effort-to-fix vulnerability class that existing reconnaissance tools universally ignore. Netra's A-F grading provides immediate actionable insight without manual analysis. The compound risk correlation engine addresses Gu and Ye's [1] explicitly stated limitation of missing cross-module intelligence, demonstrating that individually moderate findings can collectively constitute critical exposures.

Current limitations include Shodan API dependency for full network intelligence, threat intelligence API rate limits restricting enrichment to 50 subdomains per scan, and content classification based on URL patterns which may be evaded. Future work will incorporate machine learning for false positive filtering, additional threat intelligence providers, DNS zone transfer attempts, TLS certificate chain analysis, and MITRE ATT&CK technique mapping addressing the gap currently covered only by reconCTI [9].

The dual-use nature of Netra is acknowledged. Shrivas et al. [14] establish that OSINT tools are inherently double-edged instruments. Netra incorporates authorized-use enforcement through explicit disclaimer documentation and ethical usage guidelines, restricting application to owned infrastructure or explicitly authorized security assessments.

7. CONCLUSION

This paper presented Netra, a comprehensive Python-based automated reconnaissance framework integrating nine intelligence collection modules, five-provider real-time threat intelligence aggregation, HTTP security header analysis, cross-module compound risk correlation, and scan resume capability. Netra directly addresses the limitations explicitly stated by Gu and Ye [1], the most closely related prior work, while adding threat intelligence integration and header analysis capabilities absent from all tools in the comparison set.

The framework's weighted threat scoring formula, combining VirusTotal, AbuseIPDB, AlienVault OTX, URLhaus, and Feodo Tracker intelligence, transforms raw reconnaissance data into actionable threat intelligence unavailable from any existing single-purpose tool. The cross-module correlation engine identifies compound CRITICAL risks undetectable by individual modules, providing intelligence quality that directly reduces analyst time required for manual finding prioritization. Netra contributes a deployable, extensible platform for proactive attack surface assessment aligned with the defensive OSINT strategies of Avrahami et al. [4] and the compound risk reasoning principles of Atighetchi et al. [7].

REFERENCES

[1] X. Gu and J. Ye, "Fingerprint-based Vulnerability Scanning System," Procedia Computer Science, vol. 259, pp. 727-736, 2025. DOI: 10.1016/j.procs.2025.04.024

[2] Y. Liu, C. Tantithamthavorn, and L. Li, "Protect Your Secrets: Understanding and Measuring Data Exposure in VSCode Extensions," arXiv:2412.00707, Dec. 2024.

[3] P. An, R. Shafi, T. Mughogho, and O. A. Onyango, "Multilingual Email Phishing Attacks Detection using OSINT and Machine Learning," arXiv:2501.08723, Jan. 2025.

[4] Z. Avrahami, M. Zwilling, and C. Hajaj, "Leveraging OSINT for Advanced Proactive Cybersecurity Strategies and Solutions," IEEE Access, 2025.

[5] R. Ghosh, S. De, and M. Mondal, "I wasn't sure if this is indeed a security risk: Data-driven Understanding of Security Issue Reporting in GitHub Repositories of Open Source npm Packages," arXiv:2506.07728, Jun. 2025.

[6] M. S. M. S. Annamalai, E. De Cristofaro, and I. Bilogrevic, "Beyond the Crawl: Unmasking Browser Fingerprinting in Real User Interactions," in Proc. ACM Web Conference (WWW), Sydney, Australia, 2025, pp. 3896-3907.

[7] M. Atighetchi, N. Soule, R. Watro, and J. Loyall, "The Concept of Attack Surface Reasoning," in Proc. International Conference on Intelligent Systems (INTELLI), Jun. 2014. DOI: 10.13140/2.1.1491.6484

[8] F. Ahmed, P. Khatri, G. Surange, and A. Agrawal, "SearchOL: A Tool for Reconnaissance," Journal of Network and Innovative Computing, vol. 11, pp. 021-029, 2023.

[9] M. M. Rahman, S. Memon, T. Ahmed, and A. Al-Nemrat, "reconCTI: A Proactive Approach to Cyber-Threat Intelligence," in Proc. IEEE FICAC, 2025.

[10] V. Kadam and J. R. Mahajan, "Dark Reaper - Automated OSINT Tool," International Journal of Research Publication and Reviews, vol. 6, no. 10, pp. 4051-4055, Oct. 2025.

[11] R. Nealon, "Beneath the Mask: Can Contribution Data Unveil Malicious Personas in Open-Source Projects?," arXiv:2508.13453, Aug. 2025.

[12] S. Szymoniak, K. Foks, and A. Pyrkosz-Dziubczynska, "Application of OSINT Methods in Ensuring Cybersecurity," IPSI Transactions on Internet Research, 2025.

[13] J. Reza, M. I. Khan, and S. A. Sarna, "Proactive Cyber Threat Detection Using AI and Open-Source Intelligence," Journal of Computer Science and Technology Studies (JCSTS), 2025.

[14] N. Shrivas et al., "OSINT: A Double-edged Sword," in Proc. G-CARED 2025 - First Global Conference on AI Research and Emerging Developments, Greater Noida, India, 2025.

[15] S. Eskandarian, E. Messeri, J. Bonneau, and D. Boneh, "Certificate Transparency with Privacy," arXiv:1703.02209, 2017.

[16] P. Kuhn, D. N. Relke, and C. Reuter, "Common Vulnerability Scoring System Prediction based on Open Source Intelligence Information Sources," Computers & Security, vol. 131, 2023.

[17] Y. Chen et al., "Unveiling Security Vulnerabilities in Git Large File Storage Protocol," in Proc. ACM AsiaCCS, 2025.

[18] R. Williams and F. Koch, "A Scan-Based Analysis of Internet-Exposed IoT Devices Using Shodan Data," arXiv:2602.15263, Feb. 2025.