ТЕХНИКАЛЫҚ ҒЫЛЫМДАР
ТЕХНИЧЕСКИЕ НАУКИ
TECHNICAL SCIENCE

CENTRAL ASIAN SCIENTIFIC JOURNAL
ISSN 3135-3061
2026. No. 4 (32), Vol. 2. pp. 265–275
DOI: https://doi.org/10.5281/zenodo.24185732
Journal website: https://cajournal.kz

UDC 51

DEVELOPMENT OF A NETWORK SCAN DETECTION SYSTEM COMBINING SEQUENTIAL HYPOTHESIS TESTING WITH LONG-WINDOW AND TARGET-CENTRIC CORRELATION

Ilzat Khanzatov, Amir Zhumatayev, Akezhan Serkebayev
Institute of Automation and Information Technology, Satbayev University, Almaty, Kazakhstan

Ilzat Khanzatov — undergraduate student, Information Security.
Amir Zhumatayev — undergraduate student, Information Security.
Akezhan Serkebayev — undergraduate student, Information Security.
Scientific supervisor: Abdullayeva A.S.

© Ilzat Khanzatov, Amir Zhumatayev, Akezhan Serkebayev, 2026

Abstract: Network scanning is the reconnaissance step that precedes most intrusions, yet scans that are slow, low-volume or distributed across many sources often evade the fixed-threshold detectors used in practice. This paper evaluates eight open-source scan detection tools against eight functions derived from MITRE ATT&CK techniques T1046, T1595 and T1018, using an ordinal 0–2 scale and a weighted scoring model. Snort 3 obtained the highest weighted score (76.25 of 100), followed by Security Onion (72.50), while slow-scan detection (31.25% mean coverage) and distributed-scan detection (18.75%) were the weakest functions. To close these gaps, we propose NSDS, a network scan detection system that combines a threshold random walk over new destination–port pairs with evidence ageing, a 60 min distinct-failure counter and a target-centric correlation module, and maps detections to a transparent risk score. In a simulation of a network with 464 active sources and five scan scenarios (150 runs of 2 h each), NSDS detected every scan campaign, including slow and distributed scans missed by a fixed-threshold detector and by host-level TRW, while flagging on average 0.03 benign sources per run, compared with 2.04 and 5.73 for the baselines. Validation on real traffic remains future work.

Keywords: port scanning; intrusion detection; network reconnaissance; threshold random walk; distributed scanning; MITRE ATT&CK; risk scoring; network security monitoring

1. Introduction

Network scanning is the systematic probing of address ranges and service ports to discover live hosts and exposed services. It is usually the first observable stage of an intrusion and has been a constant feature of Internet traffic for decades [1]. MITRE ATT&CK describes it as Active Scanning (T1595) during reconnaissance and as Network Service Discovery (T1046) and Remote System Discovery (T1018) after initial access, when an adversary maps the internal network for lateral movement [2–4]. Open tools such as Nmap [5], ZMap [6] and Masscan [7] make scanning cheap and fast: ZMap can probe the entire IPv4 address space on one port in under an hour, and Internet-wide scanning by researchers and attackers alike is now routine [8].

Detecting reconnaissance early gives defenders time to react before exploitation, and continuous monitoring is part of the Detect function of the NIST Cybersecurity Framework 2.0 [9]. Intrusion detection and prevention systems (IDPS) are the standard control for this purpose [10]. However, most deployed detectors rely on a simple rule: an alert is raised when one source contacts

more than *N* ports or hosts within *T* seconds [11,12]. Such rules are easy to evade by slowing the scan down, by spreading it over many source addresses or by randomizing the probe order, and they also produce false alarms for legitimate hosts that contact many services, such as monitoring and backup servers [13,14].

Open-source tools remain the main option for small and medium-sized organizations, university networks and security operations centers (SOCs), including those in Kazakhstan, that cannot afford commercial network detection and response platforms. These tools differ widely in the scan types they recognize and in how they respond, and their coverage of evasive scans has not been compared systematically.

This paper addresses that problem. Its contributions are as follows: (1) a structured comparison of eight open-source scan detection tools against eight functions derived from MITRE ATT&CK, combined with a weighted scoring model; (2) the identification of the functions that are weakly covered by these tools; (3) the architecture of NSDS, a network scan detection system that combines three complementary detection modules with an explicit risk-scoring model and ATT&CK-labeled events; and (4) a reproducible simulation study that compares NSDS with a fixed-threshold detector and the original threshold random walk (TRW) on fast, slow and distributed scans. Section 2 reviews related work and states the problem, Section 3 describes the methodology, Section 4 presents the results, Section 5 discusses them, and Section 6 concludes the paper.

2. Related Work and Problem Statement

2.1. Scanning Techniques and Taxonomies

Scans are commonly classified by their footprint [11,12]. A vertical scan probes many ports on one host, a horizontal scan (sweep) probes one port across many hosts, and a block scan combines both. Scanners also differ in probe type (TCP SYN, connect, FIN, NULL, Xmas, UDP and ICMP), in rate and in coordination: a distributed or coordinated scan splits the target space among several sources so that each source stays below detection thresholds [14]. Nmap provides timing templates from T0 to T5, and its sneaky template (T1) waits 15 s between probes [5], while ZMap and Masscan randomize the order of targets [6,7]. In ATT&CK terms, external scanning maps to T1595, with sub-techniques for scanning IP blocks, vulnerability scanning and wordlist scanning, whereas internal discovery after a compromise maps to T1046 and T1018 [2–4]. Longitudinal measurements show that scanning traffic is ubiquitous and is concentrated on a small number of popular services [1,8].

2.2. Scan Detection Methods

The earliest network intrusion detection systems, Snort [15] and Bro (now Zeek) [16], detected scans by counting distinct destinations or ports per source within a time window. Staniford et al. proposed SPICE, which scores packets by their anomaly and correlates them over long periods to find stealthy portscans [13]. Jung et al. reformulated scan detection as a sequential hypothesis test: Threshold Random Walk (TRW) observes whether the first connection attempt from a remote source to each new local host succeeds or fails and applies Wald's sequential probability ratio test [17] to decide between the hypotheses "benign" and "scanner" [18]. TRW can detect scanners after as few as four or five attempts, much faster than threshold rules, and was later adapted for containing scanning worms in hardware [19]. Gates showed that coordinated scans require analysis from the target's perspective, by grouping the sources whose probes jointly cover the target space [14]. Anomaly-based approaches detect scans as changes in the entropy of traffic features [20], and machine learning models are commonly trained on labeled datasets such as CIC-IDS2017 and UNSW-NB15, which include port scan traffic [21,22]. However, dataset surveys note that such benchmarks age quickly and reflect the scanning behavior of their capture period [23].

2.3. Open-Source Detection Tools

In practice, scan detection is delivered by open-source tools. Snort 3 includes a port_scan inspector that tracks negative responses and connection counts and recognizes port scans, sweeps, decoy scans and scans of one host by many sources at configurable sensitivity levels [24]. Suricata has no dedicated scan inspector and relies on signature rules with thresholds [25]. Zeek ships a

policy script that reports address scans and port scans when a source accumulates failed connections to many hosts or ports within a configurable interval [26]. PSAD analyzes iptables log messages, assigns danger levels to scanning sources and can block them automatically [27]; scanlogd detects scans with fixed thresholds and logs them to syslog [28]; and PortSentry listens on unused ports and blocks the hosts that connect to them [29]. Wazuh correlates firewall and IDS logs with rules and can respond actively [30], while Security Onion integrates Suricata and Zeek with dashboards for analysts [31]. The scan detection capabilities of these tools are documented separately and are rarely compared.

2.4. Problem Statement

We found no lightweight open design that (a) detects fast, slow and distributed scans with one consistent model, (b) remains silent for legitimate high-fan-out hosts without manual allow-lists, (c) prioritizes detections with a transparent risk score, and (d) labels and exports them in a form usable by existing SIEM/SOC tools. This study therefore addresses three research questions: RQ1—to what extent do current open-source tools cover the core scan detection functions; RQ2—which functions are least covered; and RQ3—can a combination of sequential testing, long-window counting and target-centric correlation detect slow and distributed scans without increasing the number of false alarms?

3. Materials and Methods

3.1. Research Design

The study consisted of four stages: (i) selection of open-source scan detection tools; (ii) functional assessment and weighted scoring of the selected tools; (iii) design of the proposed system, NSDS, based on the gaps identified in stage (ii); and (iv) evaluation of NSDS against two baseline detectors in a controlled simulation.

3.2. Selection of Tools

A tool was included if (1) it is open source, (2) its documentation describes scan detection explicitly, and (3) the resulting sample covers packet-based sensors, log-based detectors and integrated monitoring platforms. Eight tools met these criteria (Table 1). PortSentry is no longer developed but was retained as a reference point for host-based detection with automatic blocking.

Table 1. Open-source scan detection tools selected for the comparative analysis (based on public documentation).

Tool	Developer	Detection approach	Status
Snort 3	Cisco Talos	Packet inspection; port_scan inspector	Active
Suricata	OISF	Packet inspection; signature rules with thresholds	Active
Zeek	Zeek Project	Connection logs; scan policy script	Active
PSAD	M. Rash (cipherdyne)	Analysis of iptables logs; danger levels	Open source
scanlogd	Openwall	Packet capture; fixed thresholds	Open source
Wazuh	Wazuh	Log correlation rules; active response	Active
Security Onion	Security Onion Solutions	Platform: Suricata, Zeek and dashboards	Active
PortSentry	Psionic Software	Listener on unused ports; blocking	Legacy

3.3. Functional Assessment

Eight functions were derived from the scan types described in Section 2.1 and from ATT&CK techniques T1046, T1595 and T1018 [2–4]: F1, real-time detection; F2, vertical scan detection; F3, horizontal sweep detection; F4, slow (low-rate) scan detection; F5, distributed or

© Central Asian Scientific Journal, 2026

decoy scan detection; F6, automatic response (blocking); F7, SIEM integration and structured export; and F8, visualization. Each tool j received an ordinal score $x_{jk} \in \{0, 1, 2\}$ for each function k (0, absent; 1, partially implemented or requires custom rules; 2, fully implemented out of the box). The scores were assigned jointly by the authors based on the official documentation available in November 2026 [24–31].

3.4. Weighted Scoring Model

Because evasive scans are the main weakness of practical detectors, the functions were grouped into six criteria with unequal weights, and a weighted score was computed for each tool as follows:

(1)

where $\bar{x}_{jc}$ is the mean score of tool j over the functions in criterion c, and w_c are the criterion weights, which sum to one. The weights were set as follows: evasive scan detection, 0.25 (mean of F4 and F5); basic scan-type coverage, 0.20 (mean of F2 and F3); real-time detection, 0.15 (F1); automatic response, 0.15 (F6); SIEM integration, 0.15 (F7); and visualization, 0.10 (F8). The coverage of each function across the tools was computed as follows:

(2)

where $N = 8$ is the number of tools. To check the robustness of the ranking, it was recomputed with equal weights for the eight functions, which is equivalent to normalizing the raw sum of scores.

3.5. Detection Model of NSDS

NSDS processes connection events $e = (t, src, dst, port, outcome)$, where the outcome is a success if the target answered (for example, with SYN-ACK for TCP) and a failure otherwise (RST, ICMP unreachable or timeout). Only events whose destination lies in the protected address space are analyzed, following [18]. The Detection Engine combines three modules.

Module A applies a threshold random walk to the first contact of each source with every new destination–port pair, rather than with every new host as in the original TRW [18]. For each source, the log-likelihood ratio is updated as follows:

(3)

where $Y_n = 1$ denotes a failed first contact, $P(Y_n = 1 \mid H_0) = 1 - \theta_0$ and $P(Y_n = 1 \mid H_1) = 1 - \theta_1$, and $\theta_0 = 0.8$ and $\theta_1 = 0.2$ are the probabilities that a first contact succeeds for a benign source and for a scanner, respectively. A source is declared a scanner when $\Lambda_n \geq \ln(P_D/P_F)$, with $P_D = 0.99$ and $P_F = 0.01$; when $\Lambda_n \leq \ln((1 - P_D)/(1 - P_F))$, it is declared benign so far and Λ_n is reset to zero. The factor $2^{-\Delta t/\tau}$, applied only to positive values of Λ, ages the accumulated evidence with a half-life τ = 10 min, so that sporadic connection errors spread over hours do not add up. Working with pairs instead of hosts makes the test sensitive to vertical scans, which the host-level TRW cannot see because they involve a single destination.

Module B counts the distinct destination–port pairs with failed first contacts per source in a sliding 60 min window and raises an alert at 20 pairs; it catches slow scans whose probes are spaced too widely for Module A. **Module C** aggregates failed probes per destination in a 5 min window. When at least 30 distinct ports of one destination fail from at least three sources, every source that contributed two or more of these ports is attributed to a coordinated campaign. This module targets distributed scans in which each source sends too few probes to be detected on its own [14].

3.6. Simulation Setup

The detectors were evaluated in a discrete-event simulation written in Python. The simulated organization had a server segment of 64 addresses (40 live servers with ports 22, 80, 443, 445, 3389 or 3306 open in different combinations), 60 workstations, a public web server, a monitoring server that polls 20 servers on three ports every minute, a backup server that opens SSH sessions to 30 servers every 15 min, two misconfigured clients that retry a dead service every 5 s, and 400 external web clients. Each workstation connected to eight preferred servers on average every 8 s, and 0–0.4% of its connections went to wrong hosts or closed ports. One scan campaign was injected into each 2 h trace: S1, a fast vertical scan of 1000 ports at 200 probes/s; S2, a fast horizontal sweep of port 445 over 254 addresses at 50 probes/s; S3, a slow vertical scan of the 100 most common ports

with 15 s between probes, as in the Nmap T1 template [5]; S4, a slow horizontal sweep of ports 22 and 3389 over the server segment with 20 s between probes; and S5, a distributed low-volume scan in which 50 sources each sent three probes to one server within 10 min. Each scenario was repeated 30 times with different random seeds, which gave 150 traces with about 50,500 analyzed events each.

NSDS was compared with two baselines: a fixed-threshold detector (THR), which raises an alert when a source contacts at least 15 distinct destination–port pairs within 60 s and approximates the counting logic of classic tools [15,28], and the original host-level TRW with the parameters given above and without ageing [18]. The metrics were the detection rate (the share of runs in which at least one scanning source was flagged), the detection delay (the number of scan probes sent before the first alert), the share of scanning sources attributed to the campaign in S5 and the number of benign sources flagged per run.

4. Results

4.1. Functional Comparison

Table 2 summarizes the functional scores. Snort 3 achieved the highest weighted score (76.25), followed by Security Onion (72.50); both tools reached 12 of 16 points. PSAD, Wazuh and Zeek formed the middle group (63.75 each), Suricata scored 47.50, the legacy PortSentry scored 42.50 and scanlogd scored 32.50. The weighted ranking is shown in Figure 1.

Table 2. Functional scores of the tools (0, absent; 1, partial; 2, full), total score and weighted score S_j (Equation (1)). F1, real-time detection; F2, vertical scan; F3, horizontal sweep; F4, slow scan; F5, distributed scan; F6, automatic response; F7, SIEM integration; F8, visualization.

Tool	F1	F2	F3	F4	F5	F6	F7	F8	Sum	S_j
Snort 3	2	2	2	1	2	1	2	0	12	76.25
Security Onion	2	2	2	1	1	0	2	2	12	72.50
PSAD	2	2	1	1	0	2	1	1	10	63.75
Wazuh	1	1	1	1	0	2	2	2	10	63.75
Zeek	2	2	2	1	0	1	2	0	10	63.75
Suricata	2	1	1	0	0	1	2	0	7	47.50
PortSentry (legacy)	2	1	0	0	0	2	1	0	6	42.50
scanlogd	2	2	0	0	0	0	1	0	5	32.50
Coverage C_k, %	93.75	81.25	56.25	31.25	18.75	56.25	81.25	31.25		

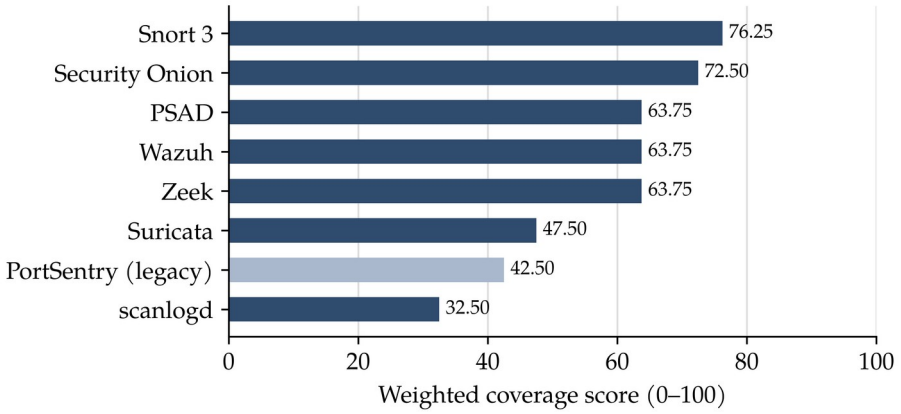


Figure 1. Weighted coverage scores of the eight scan detection tools (Equation (1)). The lighter bar denotes a legacy tool that is no longer developed.

Weighting separated the two leaders, which had equal raw totals: Snort 3 ranked first because it fully covers distributed and decoy scans, which contribute to the most heavily weighted criterion, whereas Security Onion earns its points from visualization but has no automatic response. With equal weights, Snort 3 and Security Onion were tied (75.00), PSAD, Wazuh and Zeek were

tied at 62.50, and the order of the remaining tools did not change. The leading group was therefore stable with respect to the choice of weights.

4.2. Coverage Gaps

Figure 2 shows the coverage of each function across the tools (Equation (2)). Real-time detection (93.75%; fully implemented in seven of eight tools), vertical scan detection (81.25%) and SIEM integration (81.25%) are mature. In contrast, slow-scan detection (31.25%) was not fully implemented by any tool, because all of them rely on fixed time windows that a patient scanner can outlast, and distributed-scan detection (18.75%) was fully implemented only by Snort 3. Horizontal sweep detection (56.25%) and automatic response (56.25%) were fully covered by three tools each, and visualization (31.25%) only by the two platforms. No tool reached the maximum score for all eight functions.

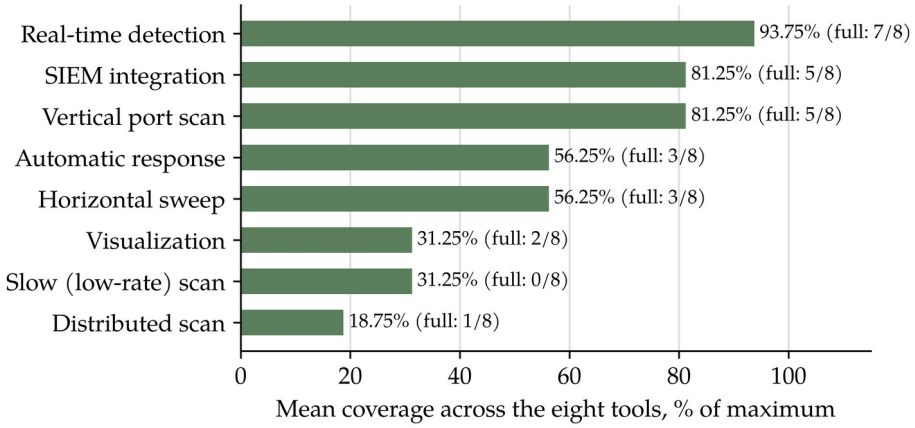


Figure 2. Coverage of the eight functions across the tools (Equation (2)); the numbers in parentheses show how many tools fully implement each function.

4.3. Proposed Architecture: NSDS

To address RQ3, NSDS is designed as a pipeline of specialized modules rather than as a monolithic sensor (Figure 3). Every module writes normalized events to a shared event store, so the chain from observation to decision remains auditable.

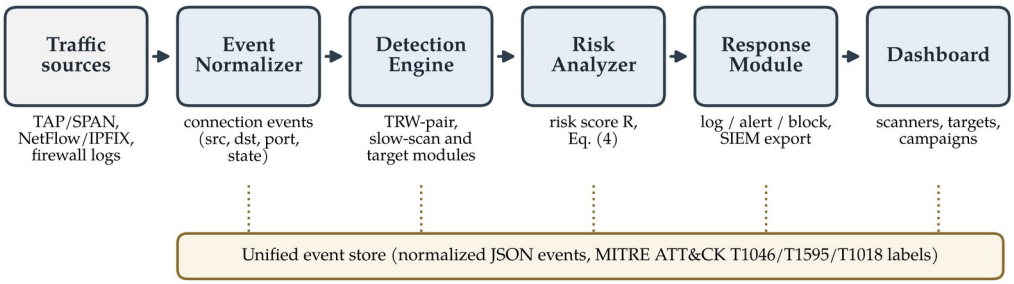


Figure 3. Architecture of NSDS. Solid arrows show the processing pipeline; dotted lines show how events are written to the unified event store.

The **Traffic sources** layer accepts packet data from a network TAP or SPAN port, flow records exported via IPFIX [32] and firewall logs. The **Event Normalizer** converts them into connection events with a source, destination, port, protocol, outcome and timestamp. The **Detection Engine** runs Modules A–C (Section 3.5) and, optionally, signature rules for probe types such as FIN, NULL and Xmas scans. Each finding is labeled with a scan type and an ATT&CK technique, as listed in Table 3.

Table 3. Mapping of scan types to MITRE ATT&CK techniques, detection modules of NSDS and default severity *s*.

Scan type	ATT&CK technique	Detection module	*s*
Vertical port scan	T1046	Module A (TRW over pairs)	0.6
Horizontal sweep (external)	T1595.001	Module A	0.7
Host discovery inside the network	T1018	Module A, ICMP echo counter	0.7
Slow (low-rate) scan	T1046, T1595.001	Module B (60 min window)	0.8
Distributed or decoy scan	T1595.001	Module C (target-centric)	0.9
Stealth probes (FIN, NULL, Xmas)	T1046	Signature rules	0.8
UDP scan	T1046	Module A on ICMP port unreachable	0.6
Vulnerability scanning	T1595.002	Signature rules, HTTP error rate	0.8
Wordlist scanning (web content)	T1595.003	HTTP 404 rate per source	0.6

The **Risk Analyzer** computes a risk score for each detection:

(4)

where $p \in [0, 1]$ is the confidence of the module (0.99 for Module A, given P_D; 0.80 for Module B; 0.85 for Module C), s is the severity of the scan type (Table 3), $c \in [0, 1]$ is the criticality of the scanned asset, and $o = \min(1, n_{open}/5)$ is the reconnaissance success, where n_{open} is the number of open services that the source has discovered. The **Response Module** maps the score to a graded response: sources with $R \geq 70$ are blocked through the firewall API for a configurable period, sources with $40 \leq R < 70$ raise an alert, and sources with lower scores are only logged. Alerts are exported to a SIEM as syslog messages [33] or as JSON, and the **Dashboard** displays scanners, scanned assets and coordinated campaigns.

Two examples illustrate the model. First, consider the slow vertical scan S3 against a database server, detected by Module A with $p = 0.99$, $s = 0.8$, $c = 0.9$ and four discovered services ($o = 0.8$). This gives $R = 82.17$, so the source is blocked. Second, consider a fast sweep of port 445 over a workstation segment, with $p = 0.99$, $s = 0.7$, $c = 0.3$ and at least five responding hosts ($o = 1.0$). This gives $R = 63.36$, so an alert is raised for an analyst.

4.4. Simulation Results

Table 4 and Figure 4 summarize the simulation. NSDS detected the scan in all 150 runs. The fixed-threshold detector detected only the two fast scans, S1 and S2, after 15 probes, and missed every slow and distributed scan, because none of them exceeded 15 targets per minute from one source. The host-level TRW detected the horizontal scans S2 and S4 but missed the vertical scans S1 and S3, which involve a single destination host, and the distributed scan S5, in which each source made only three probes. NSDS detected S1–S3 after four probes, the minimum number of consecutive failures needed to cross the upper TRW threshold, and S4 after a median of 11 probes (200 s), slightly earlier than the host-level TRW (12 probes). The distributed scan S5 was detected by Module C after a median of 34 probes (136 s), and on average 97.1% of the 50 participating sources were attributed to the campaign.

Table 4. Simulation results over 30 runs per scenario: detection rate, % of runs, with the median detection delay in probes in parentheses. The last row gives the mean number of benign sources flagged per 2 h run.

Scenario	Probes	THR	TRW	NSDS
S1, fast vertical scan	1000	100 (15)	0 (–)	100 (4)
S2, fast horizontal sweep	254	100 (15)	100 (4)	100 (4)
S3, slow vertical scan	100	0 (–)	0 (–)	100 (4)

Scenario	Probes	THR	TRW	NSDS
S4, slow horizontal sweep	128	0 (–)	100 (12)	100 (11)
S5, distributed low-volume scan	150	0 (–)	0 (–)	100 (34)
False alarms per run	–	2.04	5.73	0.03

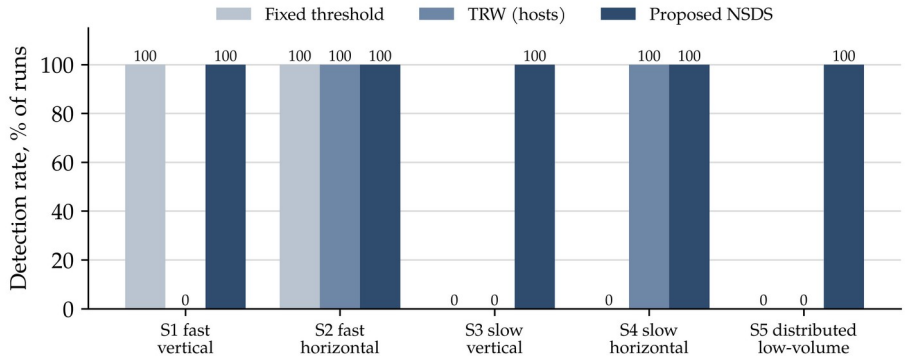


Figure 4. Detection rate of the three detectors in the five scan scenarios (30 runs per scenario).

False alarms differed more than detection rates. The threshold detector flagged the monitoring and backup servers in every run (2.04 benign sources per run on average), because legitimate polling of many services is indistinguishable from a fast scan when only targets are counted. The host-level TRW flagged 5.73 workstations per run on average, because a few failed connections to wrong hosts, accumulated over two hours, eventually crossed the upper threshold. NSDS raised false alarms in only 4 of 150 runs (0.03 benign sources per run): the monitoring and backup servers were never flagged because their first contacts succeed, and evidence ageing prevented sporadic errors from accumulating. The Python prototype processed about 0.55 million events per second on one core of a 2.1 GHz Intel Xeon processor, which is sufficient for flow-level monitoring of a medium-sized network.

5. Discussion

The comparison answers RQ1 and RQ2. Fast vertical scans are detected by almost every tool, so the practical differences lie in evasive scans, automatic response and integration. Slow and distributed scans are the weakest functions because counting within a fixed window involves an inherent trade-off: a longer window catches slower scans but also accumulates more legitimate activity, and per-source counting cannot see a campaign that is split across sources [12,14]. Packet-based sensors such as Snort 3 and Zeek scored higher on detection, whereas log-based tools such as PSAD and Wazuh scored higher on automatic response.

The simulation answers RQ3. Sequential testing over destination–port pairs removes the blind spot of the host-level TRW for vertical scans and keeps its key property: the decision depends on the ratio of failed to successful first contacts rather than on the probe rate, so a scan slowed to one probe every 15 s is detected after as few probes as a fast one [18]. Evidence ageing and the separation of failures from successes keep legitimate high-fan-out hosts silent without manual allow-lists. Target-centric correlation is necessary for low-volume distributed scans, which no per-source method can detect. The main difference between NSDS and the evaluated tools is therefore not a single better algorithm but the combination of complementary modules with a transparent risk score that can be inspected, tuned and explained to SOC analysts.

This study has several limitations. The tool scores are based on documentation rather than on hands-on testing, the ordinal 0–2 scale is coarse, and the weights reflect the authors' judgment, although the sensitivity check showed that the leading group was stable. The detectors were evaluated on synthetic traffic generated by a simple model, which cannot reproduce the diversity of real networks; for example, peer-to-peer applications, scanners operated by administrators and hosts behind network address translation would require allow-lists or additional context. An adversary who spaces probes several minutes apart or interleaves successful connections to new services can still evade Modules A and B, and Module C needs at least three sources and 30 ports per target

within its window. The parameters were not optimized, and results on real traffic, such as the port scan sessions of CIC-IDS2017 [21], may differ.

6. Conclusions

This paper compared eight open-source network scan detection tools using an ATT&CK-derived functional model and a weighted scoring model. Snort 3 showed the broadest coverage, none of the tools fully covered all eight functions, and slow-scan and distributed-scan detection were the weakest functions. To address these gaps, we proposed NSDS, which combines a threshold random walk over destination–port pairs with evidence ageing, a long-window failure counter and target-centric correlation, and maps detections to a transparent risk score with graded responses and SIEM export. In a simulation of 150 traces, NSDS detected all fast, slow and distributed scans with a median delay of 4 to 34 probes and almost no false alarms, whereas a fixed-threshold detector and the host-level TRW each missed three of the five scenarios. Future work includes validating NSDS on public datasets and real network traffic, implementing it as a Zeek or Suricata extension and tuning its parameters automatically.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data supporting the findings of this study are contained within the article (Tables 2 and 4). The simulation code is available from the corresponding author on reasonable request.

Acknowledgments: During the preparation of this manuscript, the authors used generative AI tools for drafting assistance and language editing. The authors reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ATT&CK	Adversarial Tactics, Techniques and Common Knowledge
CSF	Cybersecurity Framework
IDPS	Intrusion detection and prevention system
IPFIX	IP Flow Information Export
NIST	National Institute of Standards and Technology
NSDS	Network scan detection system
SIEM	Security information and event management
SOC	Security operations center
SPAN	Switched port analyzer
TAP	Test access point
TRW	Threshold random walk

References:

1. Allman, M.; Paxson, V.; Terrell, J. A Brief History of Scanning. In Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement (IMC ’07), San Diego, CA, USA, 24–26 October 2007; pp. 77–82.

2. MITRE Corporation. Active Scanning, Technique T1595—Enterprise. MITRE ATT&CK. Available online: https://attack.mitre.org/techniques/T1595/ (accessed on 10 November 2026).

3. MITRE Corporation. Network Service Discovery, Technique T1046—Enterprise. MITRE ATT&CK. Available online: https://attack.mitre.org/techniques/T1046/ (accessed on 10 November 2026).

4. MITRE Corporation. Remote System Discovery, Technique T1018—Enterprise. MITRE ATT&CK. Available online: https://attack.mitre.org/techniques/T1018/ (accessed on 10 November 2026).

5. Lyon, G.F. *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*; Insecure.Com LLC: Sunnyvale, CA, USA, 2009.

6. Durumeric, Z.; Wustrow, E.; Halderman, J.A. ZMap: Fast Internet-Wide Scanning and Its Security Applications. In Proceedings of the 22nd USENIX Security Symposium, Washington, DC, USA, 14–16 August 2013; pp. 605–620.

© Central Asian Scientific Journal, 2026

7. Graham, R.D. MASSCAN: Mass IP Port Scanner. Available online: https://github.com/robertdavidgraham/masscan (accessed on 10 November 2026).
8. Durumeric, Z.; Bailey, M.; Halderman, J.A. An Internet-Wide View of Internet-Wide Scanning. In Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, 20–22 August 2014; pp. 65–78.
9. National Institute of Standards and Technology. *The NIST Cybersecurity Framework (CSF) 2.0*; NIST CSWP 29; NIST: Gaithersburg, MD, USA, 2024.
10. Scarfone, K.; Mell, P. *Guide to Intrusion Detection and Prevention Systems (IDPS)*; NIST SP 800-94; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2007.
11. Bou-Harb, E.; Debbabi, M.; Assi, C. Cyber Scanning: A Comprehensive Survey. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 1496–1519.
12. Bhuyan, M.H.; Bhattacharyya, D.K.; Kalita, J.K. Surveying Port Scans and Their Detection Methodologies. *Comput. J.* **2011**, *54*, 1565–1581.
13. Staniford, S.; Hoagland, J.A.; McAlerney, J.M. Practical Automated Detection of Stealthy Portscans. *J. Comput. Secur.* **2002**, *10*, 105–136.
14. Gates, C. Coordinated Scan Detection. In Proceedings of the 16th Annual Network and Distributed System Security Symposium (NDSS 2009), San Diego, CA, USA, 8–11 February 2009.
15. Roesch, M. Snort—Lightweight Intrusion Detection for Networks. In Proceedings of the 13th USENIX Conference on System Administration (LISA ’99), Seattle, WA, USA, 7–12 November 1999; pp. 229–238.
16. Paxson, V. Bro: A System for Detecting Network Intruders in Real-Time. *Comput. Netw.* **1999**, *31*, 2435–2463.
17. Wald, A. Sequential Tests of Statistical Hypotheses. *Ann. Math. Stat.* **1945**, *16*, 117–186.
18. Jung, J.; Paxson, V.; Berger, A.W.; Balakrishnan, H. Fast Portscan Detection Using Sequential Hypothesis Testing. In Proceedings of the 2004 IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 9–12 May 2004; pp. 211–225.
19. Weaver, N.; Staniford, S.; Paxson, V. Very Fast Containment of Scanning Worms. In Proceedings of the 13th USENIX Security Symposium, San Diego, CA, USA, 9–13 August 2004; pp. 29–44.
20. Lakhina, A.; Crovella, M.; Diot, C. Mining Anomalies Using Traffic Feature Distributions. In Proceedings of the ACM SIGCOMM 2005 Conference, Philadelphia, PA, USA, 22–26 August 2005; pp. 217–228.
21. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP 2018), Funchal, Portugal, 22–24 January 2018; pp. 108–116.
22. Moustafa, N.; Slay, J. UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 10–12 November 2015; pp. 1–6.
23. Ring, M.; Wunderlich, S.; Scheuring, D.; Landes, D.; Hotho, A. A Survey of Network-Based Intrusion Detection Data Sets. *Comput. Secur.* **2019**, *86*, 147–167.
24. Cisco Talos. Snort 3 Documentation: Port Scan Inspector. Available online: https://docs.snort.org/ (accessed on 10 November 2026).
25. Open Information Security Foundation. Suricata User Guide. Available online: https://docs.suricata.io/ (accessed on 10 November 2026).
26. The Zeek Project. policy/misc/scan.zeek—Zeek Documentation. Available online: https://docs.zeek.org/en/lts/scripts/policy/misc/scan.zeek.html (accessed on 10 November 2026).
27. Rash, M. psad: Intrusion Detection and Log Analysis with iptables. Available online: https://www.cipherdyne.org/psad/ (accessed on 10 November 2026).
28. Openwall. scanlogd: A Port Scan Detection Tool. Available online: https://www.openwall.com/scanlogd/ (accessed on 10 November 2026).
29. Psionic Software. PortSentry (Sentry Tools Project). Available online: https://sourceforge.net/projects/sentrytools/ (accessed on 10 November 2026).
30. Wazuh. Wazuh Documentation: Active Response. Available online: https://documentation.wazuh.com/ (accessed on 10 November 2026).
31. Security Onion Solutions. Security Onion Documentation. Available online: https://docs.securityonion.net/ (accessed on 10 November 2026).

© Central Asian Scientific Journal, 2026

32. Claise, B.; Trammell, B.; Aitken, P. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information*; RFC 7011; Internet Engineering Task Force: Fremont, CA, USA, 2013.

33. Gerhards, R. *The Syslog Protocol*; RFC 5424; Internet Engineering Task Force: Fremont, CA, USA, 2009.

© Central Asian Scientific Journal, 2026