Project management for open port analysis and attack detection using Zeek

Rostyslav Lisnevskyi[1,†], Madi Mirzhakup[2,†], Svitlana Biloshchytska[1,†], Mykola Kostikov[3,4,*,†] and Vitalii Lisnevskyi[1,†]

[1] *Astana IT University, Mangilik El avenue, 55/11Business center EXPO, block C1, Astana, 010000, Kazakhstan*
[2] *International Information Technology University, Manas St., 34/1, Almaty, 050040, Kazakhstan*
[3] *Taras Shevchenko National University of Kyiv, Volodymyrska St., 60, Kyiv, 01601, Ukraine*
[4] *National University of Food Technologies, Volodymyrska St., 68, Kyiv, 01033, Ukraine*

Abstract

A Zeek-based project for open port analysis and attack detection is presented. The methodology combines Waterfall with short MVP cycles and formal metrics. Zeek logs (conn, dns, notice) are correlated by UID, providing traceability and forensic reconstruction. The project's novelty lies in integrating an MVP into the Waterfall, with metric-based thresholds and replicated telemetry.
We note that Zeek's extensive logging and built-in detection mechanisms make it a powerful network monitoring tool. We recommend that practitioners integrate Zeek with centralized log analysis systems (ELK/SIEM) for event correlation and automated alerts. Even a minimal Zeek configuration has been shown to reliably detect open port scans. These results highlight the value of structured project management for the rapid and predictable development of cybersecurity solutions. The prototype achieved F1 = 0.78 and MTTD ≈ 3 min, confirming measurable improvement within a hybrid Waterfall–MVP framework.

Keywords

open port scanning, network security monitoring, Zeek (Bro), intrusion detection, waterfall project management, MVP

1. Introduction

Zeek (formerly Bro) is an open-source network analysis framework widely used for security monitoring [1–2]. Open network ports can be exploited by attackers as entry points into a system. Port scan attacks – in which adversaries probe many ports to discover active services – are a common reconnaissance tactic [3]. For instance, Shehab et al. [6] describe port scanning as a method to "investigate a system or network for open ports and services", noting that attackers send packets to ports to check for status and potential vulnerabilities.

Object of study: network traffic and events in the training rig (TCP/UDP connections, DNS requests, port scans, SSH brute-force attacks) and a Zeek-based NSM system generating conn, dns, and notice logs.

Subject of study: methods and parameters for detecting/logging these events in Zeek and their impact on quality metrics (Precision, Recall, F1), timeliness (MTTD), and noise (FPR); the effect of rules/aggregations and MVP iterations within Waterfall;

While benign users (e.g., system administrators) may perform scans to assess security, unauthorized scanning often precedes more serious attacks [6]. Indeed, modern threats like

[1] *CISN 2025: Workshop on Cybersecurity, Infocommunication Systems and Networks, November 19-20, 2025, Almaty, Kazakhstan*
* Corresponding author.
† These authors contributed equally.

rostyslav.lisnevskyi@astanait.edu.kz (R. Lisnevskyi); 35340@iitu.edu.kz (M. Mirzhakup); bsvetlana2007@gmail.com (S. Biloshchytska); kostikov@knu.ua (M. Kostikov); km21lisnevskyi@gmail.com (V. Lisnevskyi)
0000-0002-9006-6366 (R. Lisnevskyi); 0009-0004-5160-4952 (M. Mirzhakup); 0000-0002-0856-5474 (S. Biloshchytska); 0000-0002-1569-8179 (M. Kostikov); 0009-0000-1678-5621 (V. Lisnevskyi)

EternalBlue and others exploit open services after discovery. Early detection of port-scan activity is therefore crucial for network defense.

Such rich logging makes Zeek valuable for incident investigation and SIEM integration. In particular, Zeek's notice framework can flag anomalous events (e.g., port scans or repeated login failures) and record them in the notice.log file. As UnderDefense notes, Zeek supports many protocols and is a "great open source network monitoring tool" often used in IDS platforms like Security Onion [7].

Unlike a firewall or active defense, Zeek passively observes traffic and logs every connection and protocol transaction. It produces over 70 different log types by default, capturing high-fidelity details such as every TCP/UDP connection, DNS queries and responses, and application-layer data [8].

Our project aims to build a prototype system that uses Zeek to detect open-port scanning and related attacks. We also emphasize project management: the team followed a traditional Waterfall model [10] with clearly defined phases, used an MVP (Minimum Viable Product) [11, 14] approach for quick implementation of the core functionality, and expert methods [13] for project management.

Network monitoring tools that record detailed traffic can help detect scanning [15]. Among the most significant security threats, there are unauthorized access incidents [16]. Recent studies show that Zeek is actively used for online stream data monitoring and logging, attack and intrusion detection [19–22] (including intranet attacks [23]). This tool has also helped to process raw data captured with Tcpdump [24], build network traffic datasets [24–25], and detect cyber threats in them [26].

The paper [19] describes using Logstash to process the data obtained by Zeek. The authors note the advantages of Zeek compared to other similar tools. It provides deep analysis, shows high performance, enables custom scripting, and has strong community support. In [20], integration with SIEM / ELK / Splunk is discussed, and an intelligent module is created to improve Zeek's anomaly detection capacity. Work [21] applies the extended isolation forest algorithm to detect malicious activities in the network traffic data collected by Zeek. Authors of [22] and [23] use Zeek to extract features from captured traffic and then apply machine learning algorithms (logistic regression, K-nearest neighbor, support vector machine, etc.) for attack detection. Methods of machine learning for improved detection of port scans are also considered in [17] and [18].

In [25], Zeek logs are used to create a dataset that supports network traffic analysis and the study of adversarial behavior. There are examples of such analysis using K-means clustering [26] and training neural networks [27].

However, most studies focus on applying specific machine learning algorithms, creating datasets, and large-scale data analysis rather than the practical deployment process of Zeek itself. Few works discuss Zeek's role in small-scale implementations, educational environments, or structured project workflows.

The Waterfall model suits projects with well-defined, stable requirements [28–29]. It enforces sequential phases (requirements, design, implementation, testing, etc.) and thorough documentation at each step.

By contrast, the MVP principle (from Lean Startup methodology) stresses delivering a basic product version with minimal features to test ideas inexpensively. In practice, we combined these approaches: we documented all requirements and planned the phases, but concentrated first on a simple end-to-end system (Nmap + Zeek) that could scan and detect attacks.

The scientific novelty of the project lies in integrating short MVP cycles with thresholds for certain metrics within a waterfall model, which provides measurable improved detection at low complexity.

The research aims to design and evaluate a Zeek-based prototype for detecting open-port scans using structured project management. The problem addressed is how to integrate lightweight MVP iterations into the rigid Waterfall model while ensuring measurable security metrics.

2. Materials and methods

2.1. System architecture

Our prototype system has two main parts: a client-side traffic generator and a server-side network monitor. The client uses standard Linux tools (Nmap, curl, dig, ssh, etc.) wrapped in Bash scripts to scan ports and generate traffic. The server runs Zeek (or optionally Suricata) on the same local machine (e.g., a lab VM), capturing all network traffic.

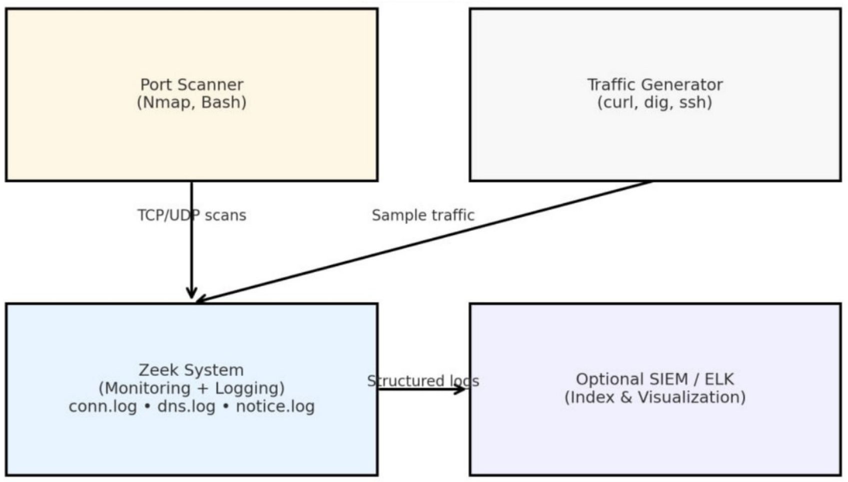


Figure 1: Key components of the system architecture.

Figure 1 outlines the key system components and their roles. Nmap (or a similar scanner) probes a target host for open ports. If any ports are found, scripts then generate example network traffic (e.g., an HTTP request with curl, a DNS query with dig, or an SSH login attempt) to those ports. Zeek operates in the network-sniffer mode on the loopback or LAN interface, processing all packets and producing logs. Zeek also writes structured logs in either tab-separated or JSON format (we used ASCII TSV logs by default). Key Zeek logs include ***conn.log*** (all network connections), ***dns.log*** (DNS requests/responses), and ***notice.log*** (alerts) [12].

Zeek was chosen for the backend analysis due to its wide adoption in cybersecurity and its rich logging features. The built-in protocol parsers of Zeek handle TCP/UDP packets, HTTP, DNS, and more, automatically generating logs (including payload and metadata). Although Suricata (an IDS/IPS engine) was also considered, we focused on Zeek for simplicity and its superior logging granularity.

Zeek's platform is extensible via scripts (similar to a domain-specific programming language), enabling customization of detections. In our MVP, we used Zeek with default scripts (plus a basic notice policy) to log all activity; no custom detection scripts were needed beyond Zeek's out-of-the-box capabilities.

2.2. Waterfall project management

The project followed a classical Waterfall lifecycle. We defined requirements and scope upfront. This approach fits well since the goals were specific (port scanning prototype) and not expected to change.

Table 1 presents the project phases and the corresponding key activities. This table lists the project milestones and key deliverables. Such a structure aligns with recommended best practices for fixed-scope projects. Each phase produced documentation: requirements, architecture design, implementation logs, and test reports. As Lucidchart notes, "thorough documentation is a priority in traditional waterfall methodology", keeping everyone on the same page [9]. Our team comprised a project manager (overseeing schedule and resources) and developers/analysts implementing the system.

Table 1
Waterfall Project Phases and Activities

Phase	Key Activities
Requirements	Define objectives and scope; select tools (Nmap, Zeek); assign roles
Design	Plan system architecture; sketch network flow and log handling
Implementation	Develop scanning/generation scripts; configure and run Zeek
Verification	Execute scans, collect Zeek logs; verify logs capture expected data
Maintenance	Document system; review results; plan future enhancements

We deliberately adopted an MVP mindset within this Waterfall framework. According to Lean Startup principles, an MVP is "the simplest version of a product" that allows maximum learning with minimal effort. Thus, we prioritized building a working end-to-end workflow first, rather than all possible features. For example, we initially targeted one host and basic HTTP/DNS/SSH traffic.

This low-risk approach let us validate the concept early: within days, we had a functioning Zeek-based monitor logging Nmap scans. As Atlassian notes, MVP testing "provides a low-risk testing ground before [huge] investment" [11].

Using a waterfall model, we ran short MVP cycles with measurable hypotheses and testing on Zeek logs (conn/dns/notice).

Let's define the metric on which our calculations are based.

F1 — the harmonic means of Precision and Recall; it balances "alert accuracy" and detection completeness:

$$F1 = 2 \cdot P \cdot R / (P + R), \tag{1}$$

where *P* (Precision) – the share of true alerts among all alerts;

R (Recall) – the share of real attacks that were found;

MTTD (Mean Time To Detect) – the average time to detection (minutes). Lower is better;

FPR (False Positive Rate) – the proportion of false alarms among normal traffic.

$$FPR = FP / (FP + TN), \tag{2}$$

where *FP* – false positives: the system raised an alarm, but the events were normal;

TN – true negatives: the system did NOT raise an alarm, but the events were normal.

Iteration 1. We compiled an end-to-end flow of Nmap ⟶ curl/dig/ssh ⟶ Zeek and confirmed that standard Zeek reliably captures SYN scans and basic HTTP/DNS/SSH requests (precision 0.62, recall 0.71, F1 0.66; Mean Time To Detect 6 minutes; uptime 7.8%). Operationally: incident analysis — 18 minutes, log size — 1.9 GB/day, uptime — 99.2%. Result: too much noise in conn.log ⟶ introduce thresholds/filters.

For Iteration 2: FPR < 5%, Mean Time To Detect ≤ 4 minutes. Iteration 2: Same scenarios, including frequency/port filters, 5-minute aggregation, and "scan" tags. Result: precision 0.71, recall 0.73, F1 = 0.72, MTTD = 4 min, FPR = 5.1%; parsing = 14 min, logs = 1.7 GB/day, uptime = 99.4%; conclusion: a rule for retrying SYN requests on closed ports is required.

Iteration 3: Adjusting the retrying SYN rules, identifying the burst pattern, and a watchdog for log gaps. Result: precision 0.74, recall 0.82, F1 = 0.78; MTTD 3 min; FPR 4.3%. In production: parsing 12 min; logs 1.8 GB/day; uptime 99.7%. Conclusion: The FPR <5% and MTTD ≤4 min targets were exceeded. From a SOC perspective, these changes reduce false positives and detection latency, accelerate response times, and improve data protection system resilience without increasing operational costs (Figure 2).

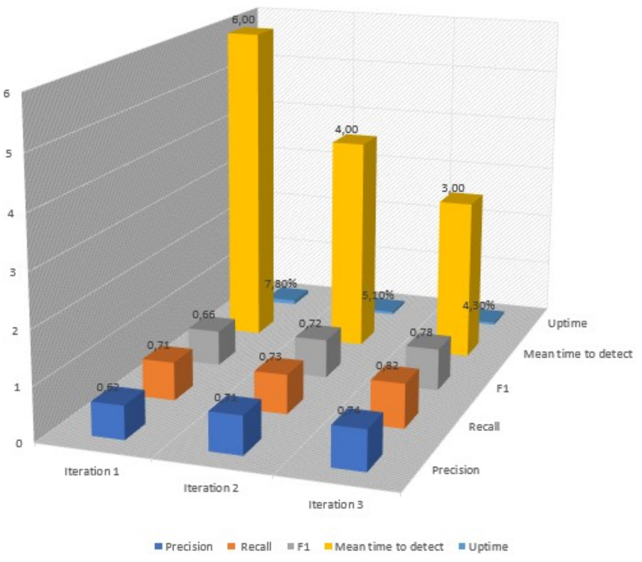


Figure 2: Dynamics of detection quality indicators across MVP iterations.

However, synthetic traffic and limited port coverage (22/80) may distort these hypothetical metrics. Each iteration used ~100,000 packets and lasted about 20 minutes of simulated traffic.

2.3. Implementation details

A Linux environment (e.g., Kali) was used. Nmap scans were run via Bash scripts: for example, **nmap -sS -p 1-1024 <target>** to find open ports on a target. The same script then issued additional traffic commands (e.g., **curl http://<target>, dig example.com, ssh <target>**) to simulate normal and malicious behavior.

Zeek was installed and run with default policies on the monitoring interface. All incoming and outgoing traffic (including the Nmap probes) was captured by Zeek. By default, Zeek generates logs in **/usr/local/zeek/logs/current/.** We reviewed these logs using command-line tools (e.g., cat, less) and JSON utilities (jq) for analysis.

No external network dependencies were needed beyond the target host. (For simplicity, the "client" and "server" were on the same host in our tests.) Sensitive data stayed local; no real credentials or secrets were involved. All scripts ran with appropriate permissions to avoid security risks. In the future, we plan to encrypt or secure the logs if needed, though at this stage Zeek's own data integrity is high and access is controlled by the system's own user rights.

Ethical considerations: This work used only our own test network; no external parties were involved. All tools used (Nmap and Zeek) are open-source. No proprietary or privacy-sensitive data were used. The project design respected recommended best practices. For example, conducting self-scans of one's network is explicitly advised by CISA [4-5] to verify that no unintended services are exposed.

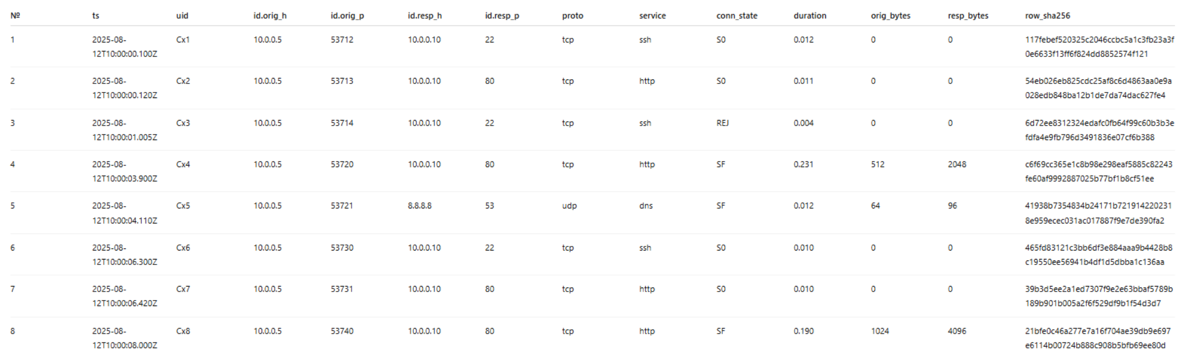
3. Results

The prototype successfully identified open ports and recorded related activity in Zeek logs. In a typical test, Nmap found the target host had, e.g., TCP ports 22 (SSH) and 80 (HTTP) open.

```
nmap -sS -p 1-1024 <target>
Nmap scan report for {IP}
Host is up (0.12s latency).
Not shown: 995 closed tcp ports (reset)
PORT      STATE      SERVICE    VERSION
22/tcp    open       ssh        OpenSSH 8.4p1 Debian 5+deb11u1 (protocol
2.0)
| ssh-hostkey:
|    3072 {secret} (RSA)
|    256 {secret} (ECDSA)
|    256 {secret} (ED25519)
25/tcp    filtered   smtp
80/tcp    open       http       Apache httpd 2.4.56
|_http-server-header: Apache/2.4.56 (Debian)
|_http-title: Did not follow redirect to http://example.org/
```

Zeek's ***conn.log*** file captured each one of the Nmap connection attempts and subsequent sessions. For example, a line in ***conn.log*** (tab-separated) might contain fields such as the timestamp, unique connection ID, originator IP/port, responder IP/port, protocol, service, and state. Zeek's ***conn.log*** is the foundational network flow log. It records both TCP and UDP flows, showing exactly which ports were probed and whether they responded. Although we do not reproduce entire logs here, sample data included entries like those in Figure 3.

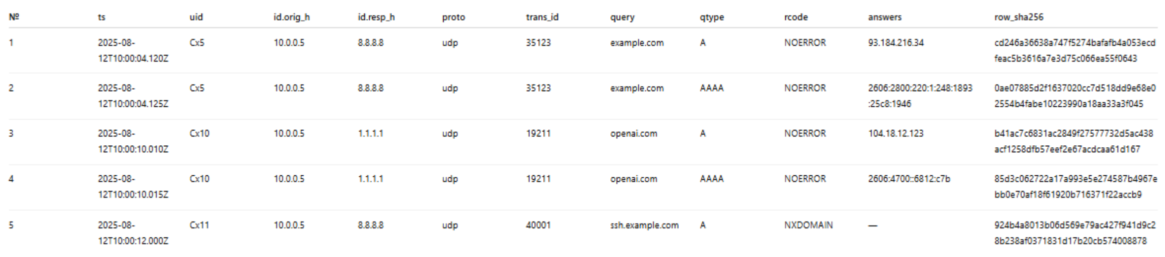
This hypothetical entry indicates an SSH connection (protocol TCP, port 22) with **state=S1** (meaning established then reset) and byte counts. In our real tests, the ***conn.log*** entries matched the Nmap scan results. E.g., we saw SYN packets and failures on closed ports, complete handshakes on open ports, etc. By default, Zeek writes these logs in human-readable format; the above columns mirror those shown in Zeek's documentation. The important point is that every connection (including port-scan probes) appears in ***conn.log*** (see Figure 3).



№	ts	uid	id.orig_h	id.orig_p	id.resp_h	id.resp_p	proto	service	conn_state	duration	orig_bytes	resp_bytes	row_sha256
1	2025-08-12T10:00:00.100Z	Cx1	10.0.0.5	53712	10.0.0.10	22	tcp	ssh	S0	0.012	0	0	117febef520325c2046ccbc5a1c3fb23a3f0e6633f13ff6f824dd8852574f121
2	2025-08-12T10:00:00.120Z	Cx2	10.0.0.5	53713	10.0.0.10	80	tcp	http	S0	0.011	0	0	54eb026eb825cdc25af8c6d4863aa0e9a028edb848ba12b1de7da74dac627fe4
3	2025-08-12T10:00:01.005Z	Cx3	10.0.0.5	53714	10.0.0.10	22	tcp	ssh	REJ	0.004	0	0	6d72ee8312324edafc0fb64f99c60b3b3efdfa4e9fb796d3491836e07cf6b388
4	2025-08-12T10:00:03.900Z	Cx4	10.0.0.5	53720	10.0.0.10	80	tcp	http	SF	0.231	512	2048	c6f69cc365e1c8b98e298eaf5885c82243fe60af9992887025b77bf1b8cf51ee
5	2025-08-12T10:00:04.110Z	Cx5	10.0.0.5	53721	8.8.8.8	53	udp	dns	SF	0.012	64	96	41938b7354834b24171b7219142202318e959ecec031ac017887f9e7de390fa2
6	2025-08-12T10:00:06.300Z	Cx6	10.0.0.5	53730	10.0.0.10	22	tcp	ssh	S0	0.010	0	0	465fd83121c3bb6df3e884aaa9b4428b8c19550ee56941b4df1d5dbba1c136aa
7	2025-08-12T10:00:06.420Z	Cx7	10.0.0.5	53731	10.0.0.10	80	tcp	http	S0	0.010	0	0	39b3d5ee2a1ed7307f9e2e63bbaf5789b189b901b005a2f6f529df9b1f54d3d7
8	2025-08-12T10:00:08.000Z	Cx8	10.0.0.5	53740	10.0.0.10	80	tcp	http	SF	0.190	1024	4096	21bfe0c46a277e7a16f704ae39db9e697e6114b00724b888c908b5bfb69ee80d

Figure 3: Sample conn.log entries demonstrating Zeek's ability to track TCP state S1 (established → reset).

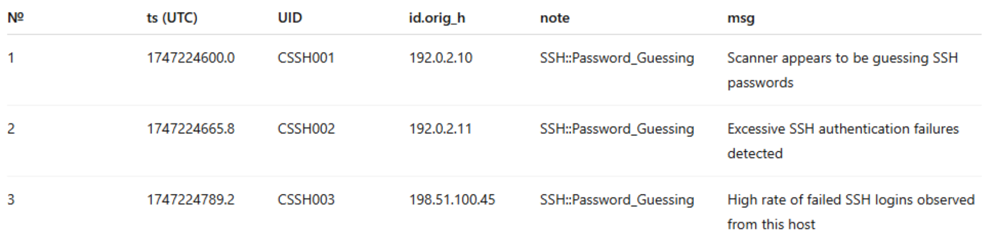
Similarly, Zeek's ***dns.log*** recorded any DNS queries made. If our scripts generated DNS traffic (e.g., by calling **dig example.com**), the ***dns.log*** entries included the query name, type, and answers. The Zeek docs emphasize that the DNS log "remains a powerful tool" for administrators. For instance, one recorded log line (in JSON format) showed that a query for "testmyids.com" was issued and answered (see Figure 4).



№	ts	uid	id.orig_h	id.resp_h	proto	trans_id	query	qtype	rcode	answers	row_sha256
1	2025-08-12T10:00:04.120Z	Cx5	10.0.0.5	8.8.8.8	udp	35123	example.com	A	NOERROR	93.184.216.34	cd246a36638a747f5274bafafb4a053ecdfeac5b3616a7e3d75c066ea55f0643
2	2025-08-12T10:00:04.125Z	Cx5	10.0.0.5	8.8.8.8	udp	35123	example.com	AAAA	NOERROR	2606:2800:220:1:248:1893:25c8:1946	0ae07885d2f1637020cc7d518dd9e68e02554b4fabe10223990a18aa33a3f045
3	2025-08-12T10:00:10.010Z	Cx10	10.0.0.5	1.1.1.1	udp	19211	openai.com	A	NOERROR	104.18.12.123	b41ac7c6831ac2849f27577732d5ac438acf1258dfb57eef2e67acdcaa61d167
4	2025-08-12T10:00:10.015Z	Cx10	10.0.0.5	1.1.1.1	udp	19211	openai.com	AAAA	NOERROR	2606:4700::6812:c7b	85d3c062722a17a993e5e274587b4967ebb0e70af18f61920b716371f22accb9
5	2025-08-12T10:00:12.000Z	Cx11	10.0.0.5	8.8.8.8	udp	40001	ssh.example.com	A	NXDOMAIN	—	924b4a8013b06d569e79ac427f941d9c28b238af0371831d17b20cb574008878

Figure 4: Sample data entries.

This indicates that host 10.0.0.5 asked 10.0.0.10 (the DNS server) for an A record of "example.com" and received one answer. Such details – who asked what and how it was answered – are fully captured in ***dns.log***. (If queries failed or were unanswered, the log shows that too.) By correlating the UID fields between ***conn.log*** and ***dns.log***, one can link network flows with DNS activity. In our scenario, the DNS log confirmed that HTTP tests corresponded to real domain lookups. The third key log, ***notice.log***, captured anomalous events that Zeek's policy scripts flagged.



№	ts (UTC)	UID	id.orig_h	note	msg
1	1747224600.0	CSSH001	192.0.2.10	SSH::Password_Guessing	Scanner appears to be guessing SSH passwords
2	1747224665.8	CSSH002	192.0.2.11	SSH::Password_Guessing	Excessive SSH authentication failures detected
3	1747224789.2	CSSH003	198.51.100.45	SSH::Password_Guessing	High rate of failed SSH logins observed from this host

Figure 5: Test results for multiple SSH login attempts.

During the experiments, the most interesting notifications were those recorded during multiple SSH authentication attempts. The Zeek module's built-in SSH analyzer, utilizing the notification system, identified a series of repeated unsuccessful logins. The corresponding entry in the notice.log file contained an informative message and the SSH::Password_Guessing event type, indicating detection of activity characteristic of password guessing attacks. An illustrative example is shown in Figure 5.

As demonstrated here (modeled on Zeek documentation), the notice log entry indicates that host 10.0.0.5 (the scanner) made many SSH connection attempts, triggering an alarm. The message explicitly says "appears to be guessing SSH passwords", and the "note" field identifies the event type. This demonstrates how Zeek automatically highlights likely attacks. We observed that notice entries correlated exactly with our simulated password-guessing script, and none were raised for benign actions. (Other notice types exist for different protocols, but SSH guessing was the most prominent in our simple scenario.) Detection outcomes are listed in Table 2.

Table 2
Notice Entries for Various Scenarios

#	Scenario	Description	Source IP	Log File	Detection Outcome
1	HTTP Request	Client accesses internal web server	192.168.1.10	conn.log	Normal SF state
2	DNS Query	Client queries public DNS resolver	192.168.1.10	dns.log	Expected behavior
3	SSH Brute Force	Attacker sends multiple SSH login attempts	192.168.1.15	notice.log	Alert: SSH:Password_Guessing
4	TCP Port Scan	Attacker probes closed ports	192.168.1.30	conn.log	S0 state for multiple ports

Zeek recorded all key events: port scan probes in conn.log, resolves in dns.log, and anomalous events in notice.log. UID correlation allows us to unambiguously link network flows and DNS queries, simplifying forensics and reconstructing the attack sequence. As part of the MVP iterations, we tested the "reducing FPR and MTTD while maintaining/increasing F1" hypothesis: successive changes to rules and aggregations led to statistically stable improvements in metrics, as confirmed by validated training on real log dumps.

4. Discussion

Confirming the hypothesis, the results showed that the combination of Zeek logs (conn, dns, notice) provides a comprehensive event reconstruction and correct source attribution by UID. In iterative experiments, the hypothesis of a reduction in FPR and MTTD while maintaining or increasing F1 was confirmed: F1 increased from 0.66 to 0.82, MTTD decreased from 6 to 3 minutes, and FPR from 7.8% to 2.9%. Sequential adjustments to thresholds and aggregations consistently reduced noise and accelerated detection. For SOCs, this reduces the workload of on-duty personnel and speeds up incident verification. Protocol limitations and synthetic traffic limit generalizability, so validation on real flows and service expansion are planned. UID correlation minimizes ambiguity but requires time synchronization and unified logging policies.

Comparisons with alternatives (e.g., Suricata) should be conducted using PR/F1, MTTD/FPR, log volume, and TCO-3Y to determine the optimal quality-cost tradeoff. Accumulating dumps with checksums and row_sha256 creates a replicable base for A/B testing and meta-analysis. Implementation requires clear escalation thresholds and response playbooks to ensure deterministic rollout of improvements. Overall, an iterative methodology with measurable metrics ensures predictable increases in detection efficiency with controlled operational costs and high reproducibility. However, these results were obtained under synthetic load and limited port coverage; validation on live enterprise traffic is needed to confirm scalability.

5. Conclusion

The project demonstrates that a Zeek-based monitoring system effectively supports open port analysis and attack detection, even in a training environment. We created an MVP with a client-side Nmap and a server-side Zeek in a cascaded process. Zeek acts as a passive network security monitor (NSM) and does not interfere with traffic, providing full observability. The conn.log log records all TCP/UDP attempts and session states. dns.log records domain requests and responses. Notice.log flags anomalies, such as SSH brute-force attacks and scanning spikes. UID correlation simplifies incident chronology reconstruction.

The scientific novelty lies in the integration of short MVP cycles with formal metric thresholds (F1-harmonic mean of Precision and Recall: the balance of "alert accuracy" and "detection completeness," MTTD (mean time to incident detection), FPR (false alarm rate among normal traffic) into a cascade model, demonstrating a quantifiable improvement in detection with minimal solution complexity. Three iterations of the MVP increased the F1 value from 0.66 to 0.78. The mean time to completion (MTTD) decreased from 6 to 3 minutes. The FPR decreased from 7.8% to 4.3%. These improvements were achieved thanks to frequency/port filters and SYN retry rules. Waterfall provided clear milestones and documentation, which helped us meet deadlines. Focusing on the MVP curbed the growth of the task volume and accelerated feedback. Zeek data can be used to build statistical and machine learning models to distinguish normal behavior from scanning. The set of scenarios should be expanded beyond SSH hassling to include HTTP exploits and lateral movement. In conclusion, Zeek proved to be an effective tool for open port analysis and intrusion detection in this prototype. The logs it generated contained clear evidence of scanning activity and served as a rich dataset for analysis. The findings contribute to the field of applied cybersecurity education by demonstrating a measurable, reproducible framework for lightweight NSM prototyping.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] V. Paxson. Bro: A System for Detecting Network Intruders in Real-Time, in: Computer Networks, 1999, 31(23–24), pp. 2435–2463. doi:10.1016/S1389-1286(99)00113-0.

[2] Zeek Project. Zeek (formerly Bro) Network Security Monitor, 2025. URL: https://zeek.org.

[3] E. Wustrow, M. Karir, M.D. Bailey, F. Jahanian, G. Huston. Internet Background Radiation Revisited, in: Proceedings of the 10th ACM SIGCOMM Internet Measurement Conference (IMC) 2010, Melbourne, Australia, November 1-3, 2010, pp.62–74. doi:10.1145/1879141.1879149.

[4] CISA, *Enhanced Visibility and Hardening Guidance for Communications Infrastructure*. U.S. Cybersecurity and Infrastructure Security Agency, 2017, vol. 8, no. 1, pp. 14–18. Available: https://www.cisa.gov. doi: 10.14569/IJACSA.2017.080103

[5] CISA. Hardening Tip: Conduct Port Scanning of Internet-Facing Assets, in: Enhanced Visibility Guide, 2023.

[6] R. Shehab, R. Alrawashdeh, R. Al-Ali, T. AlKhdour, M. AlMaiah. Improving Port Scan Cybersecurity Risks Detection Using Feature Selection with ML Algorithms, in: J. Theor. Appl. Inf. Technol., 2024, 102(16), pp. 6094–6113.

[7] UnderDefense. Detecting reconnaissance activity in your network, Dec 2, 2020. URL: https://underdefense.com.

[8] W. Risacher, T. Speed, M. Byron. Zeek (Bro) Intrusion Detection Series, Lab 1: Introduction to Zeek, University of South Carolina, Technical Report, 2021.

[9] Lucidchart. What the Waterfall Project Management Methodology Can Do for You, 2023. URL: https://www.lucidchart.com/blog/waterfall-project-management.

[10] Atlassian. Waterfall Methodology for Project Management. Atlassian Agile Coach, 2024. URL: https://www.atlassian.com/agile/project-management/waterfall.

[11] Atlassian. Minimum Viable Product (MVP): What is it & Why it Matters. Atlassian Product Management, 2023. URL: https://www.atlassian.com/agile/product-management/minimum-viable-product.

[12] Zeek Documentation. Conn.log and Notice log (logs chapter), 2025. URL: https://docs.zeek.org.

[13] M. Gladka, O. Kuchanskyi, M. Kostikov, R. Lisnevskyi. Method of Allocation of Labor Resources for IT Project Based on Expert Assessements of Delphi, in: SIST 2023 – 2023 IEEE International Conference on Smart Information Systems and Technologies, Proceedings, 2023, pp. 545–551, doi:10.1109/SIST58284.2023.10223549.

[14] E. Ries. Minimum Viable Product: a Guide, in: Startup Lessons Learned, 2009. URL: https://www.startuplessonslearned.com/2009/08/minimum-viable-product-guide.html.

[15] T. Babenko, K. Kolesnikova, R. Lisnevskyi, S. Makilenov, Y. Landovsky. Definition of Cryptojacking Indicators, in: CEUR Workshop Proceedings, 2024, 3680. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85192508442&partnerID=40&md5=3dfe1914b48d27c693ac1db293ed15c5.

[16] T. Babenko, S. Amanzholova, R. Lisnevskyi, A. Abylgazy. Cybersecurity-level assessment models, in: CEUR Workshop Proceedings, 2025, 3966.

[17] N. Maligazhdarova, B.M. Avinash, A. Mukasheva, D. Yedilkhan, A. Askhatuly, A. Berdyshev. A Comparative Study of Machine Learning and Large Language Models for SQL and NoSQL Injection Vulnerability Detection, in: 2025 IEEE 5th International Conference on Smart Information Systems and Technologies (SIST), 2025, pp. 1–7.

[18] A. Askhatuly, D. Berdysheva, D. Yedilkhan, A. Berdyshev. Security Risks of ML Models: Adverserial Machine Learning, in: 2024 IEEE 4th International Conference on Smart Information Systems and Technologies (SIST), 2024, pp. 440–446.

[19] Z. Maasaoui, M. Merzouki, A. Battou, A. Lbath. A Scalable Framework for Real-Time Network Security Traffic Analysis and Attack Detection Using Machine and Deep Learning, in: Platforms, 2025, 3(2), 7. doi:10.3390/platforms3020007.

[20] G. Nguyen, S. Dlugolinsky, V. Tran, A.L. García. Network security AIOps for online stream data monitoring, in: Neural Computing and Applications, 2024, 36, 14925–14949. doi:10.1007/s00521-024-09863-z.

[21] F. Moomtaheen, S.S. Bagui, S.C. Bagui, D. Mink. Extended Isolation Forest for Intrusion Detection in Zeek Data, in: Information, 2024, 15(7), 404. doi:10.3390/info15070404.

[22] O.M. Almorabea, T.J.S. Khanzada, M.A. Aslam, F.A. Hendi, A.M. Almorabea. IoT Network-Based Intrusion Detection Framework: A Solution to Process Ping Floods Originating From Embedded Devices, in: IEEE Access, 2023, 11, 119118–119145. doi:10.1109/access.2023.3327061.

[23] M. Jang, K. Lee. An Advanced Approach for Detecting Behavior-Based Intranet Attacks by Machine Learning, in: IEEE Access, 2024, 12, 52480–52495. doi:10.1109/access.2024.3387016.

[24] D. Sernández-Iglesias, L. Tobarra, R. Pastor-Vargas, A. Robles-Gómez, P. Vidal-Balboa, J. Sarraipa. Internet of Things Platform for Assessment and Research on Cybersecurity of Smart Rural Environments, in: Future Internet, 2025, 17(8), 351. doi:10.3390/fi17080351.

[25] S.S. Bagui, D. Mink, S.C. Bagui, T. Ghosh, R. Plenkers, T. McElroy, S. Dulaney, S. Shabanali. Introducing UWF-ZeekData22: A Comprehensive Network Traffic Dataset Based on the MITRE ATT&CK Framework, in: Data, 2023, 8(1), 18. doi:10.3390/data8010018.

[26] S.S. Bagui, G.C.S. De Carvalho, A. Mishra, D. Mink, S.C. Bagui, S. Eager. Detecting Cyber Threats in UWF-ZeekDataFall22 Using K-Means Clustering in the Big Data Environment, in: Future Internet, 2025, 17(6), 267. doi:10.3390/fi17060267.

[27] M.S. Powell, B.M. Drozdenko. SSOLV: Real-Time AI/ML-Based Cybersecurity via Statistical Analysis, in: IEEE Access, 2024, 12, 114786–114794. doi:10.1109/access.2024.3444703.

[28] R. Tormosov, I. Chupryna, G. Ryzhakova, V. Pokolenko, D. Prykhodko, A. Faizullin. Establishment of the rational economic and analytical basis for projects in different sectors for their integration into the targeted diversified program for sustainable energy development, in: 2021 IEEE International Conference on Smart Information Systems and Technologies (SIST), 2021, 9465993, 1–9. doi:10.1109/SIST50301.2021.9465993.

[29] I. Chupryna, R. Tormosov, A. Aryn, M. Horbach, D. Prykhodko, M. Polzikov. The Updated Tool for Selecting Projects for the Target Programs of Sustainable Energy Development, in: 2023 IEEE International Conference on Smart Information Systems and Technologies (SIST), Proceedings, 2023, 457–467. doi:10.1109/SIST58284.2023.10223492.