IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510117

Aegis: An AI-Based Network Intrusion Detection and Prevention System with Stateful Scan Detection and ISO/IEC 27001 Incident Reporting

M. Sreedharan[1], Lokesh S[2], Deepashri P[3], P. Durgadevi[4]

Department of Computer Science and Engineering, SRM Institute of Science and Technology, Vadapalani Campus, Chennai, India[1-4]

Abstract: Most published machine learning intrusion detectors report accuracy above 99% on CIC-IDS2017, yet much of that figure comes from duplicate flows that end up in both the training and test sets. We built an AI-based network intrusion detection and prevention system (NIDS/NIPS) for Windows hosts that pairs flow classification with a stateful scan detector, decoy honeypot ports and a monitor for failed Windows logons. Alerts are mapped to MITRE ATT&CK techniques and ISO/IEC 27001:2022 controls. Before training, we removed host identifiers and 12,537 exact duplicate flows (22.16% of the data), and fitted the scaler on the training split alone. On the cleaned data XGBoost scored 99.62% accuracy, 99.39% macro F1 and a 0.55% false positive rate, with a median latency of 0.38 ms per flow; Random Forest scored 99.25%, 97.54%, 1.22% and 35.55 ms. When we held out the whole PortScan family, both supervised models called over 89% of those flows benign, and Isolation Forest caught only 0.31%. That result is why the system handles scans with a stateful detector instead of the classifier. Detections feed a four-tier escalation policy that writes Windows Firewall rules, but only after allowlist and subnet checks, with automatic expiry, and in dry-run mode unless an administrator switches enforcement on. A pure-Python flow extractor reproduces 56 CICFlowMeter features exactly and approximates 12 more, so live capture no longer needs Java.

Keywords: Network Intrusion Detection, Intrusion Prevention, XGBoost, Isolation Forest, SHAP, Data Leakage, CIC-IDS2017, Port Scan Detection, MITRE ATT&CK, ISO/IEC 27001

I. INTRODUCTION

A. Background and Motivation

Denning's 1987 model of intrusion detection rested on a simple idea: learn what normal activity looks like and flag what departs from it [1]. In practice, most networks still lean on rule engines such as Snort [2] and Zeek, formerly Bro [3]. These work well for attacks somebody has already written a rule for, and poorly for anything else. Learning from flow statistics is the obvious alternative, and labelled datasets like CIC-IDS2017 [4] have made it easy to train classifiers that report accuracy above 99%.

We were wary of those numbers from the start. Sommer and Paxson pointed out long ago that a detector which works in the lab often fails on a real network [5]. CIC-IDS2017 itself has known labelling errors and flow-construction artefacts [6], and Arp et al. found data snooping and spurious correlations to be among the most common mistakes in security ML papers [7]. A model can look excellent simply because identical flows sit on both sides of the split, and it will say nothing useful about an attack family it never saw. Detection is also only half the job. A system that cannot respond is of limited use, and one that responds carelessly, for example by blocking the default gateway, can do more harm than the attack.
We therefore set out to build a system we could defend on both counts. The evaluation guards against duplicate leakage and includes a test on a withheld attack family. Where that test showed learning failing, we added rule-based sensors. Detection is then wired to a guarded prevention engine and an information security management (ISM) layer that describes each incident in the terms an auditor uses.

B. Research Contributions

The main contributions of this work are as follows:
(i) A leakage-aware preprocessing pipeline for CIC-IDS2017 that removes identifiers and 22.16% duplicate flows, with an automated inspector that checks for single-feature shortcuts when accuracy exceeds 99.5%.
(ii) A comparison of Random Forest, XGBoost and Isolation Forest on accuracy, false positive rate, latency and throughput, together with a zero-day experiment that withholds the entire PortScan family.
(iii) A multi-sensor detection layer that combines a pure-Python flow extractor, a stateful scan detector, decoy honeypot ports and a Windows Event ID 4625 brute-force monitor.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510117

(iv) A prevention engine that maps detections to a four-tier escalation policy and enforces Windows Firewall rules through infrastructure guards, subnet scoping, automatic expiry and a dry-run default.
(v) An ISM layer that tags each alert with MITRE ATT&CK tactics and techniques, computes a quantitative risk score with CIA impact, and exports ISO/IEC 27001:2022 incident reports.

II. LITERATURE REVIEW

Rule-based tools remain the standard in deployed systems. Snort matches packets against hand-written rules [2], while Zeek turns traffic into protocol events that policy scripts can act on [3]. Both are precise and easy to audit, but neither can catch what it was not told to look for. Port scans have received their own attention: Jung et al. applied sequential hypothesis testing to connection outcomes and could label a scanner after only a handful of probes [8]. Their work makes a point that matters for our results, namely that a scan is visible in a sequence of connections and not in any single one.
Flow-based learning took off once labelled datasets became available. CIC-IDS2017 [4] provides about 80 features computed by CICFlowMeter [9] over benign traffic and a range of attacks, including DoS, brute force, web attacks, botnets and port scans. Ring et al. [10] review this and other datasets along with their weaknesses. For tabular flow data, tree ensembles are the usual choice. Random Forest [11] is robust and needs little tuning, XGBoost [12] adds regularised gradient boosting, and Isolation Forest [13] finds outliers without any labels. SHAP [14], [15] computes exact Shapley values for tree models, which lets an analyst see which features pushed a flow towards an alert.

Several studies question how much of this carries over to practice. Sommer and Paxson argued that detectors tuned on closed datasets seldom transfer to live networks [5]. Engelen et al. found mislabelled and malformed flows in CIC-IDS2017 [6], and Arp et al. showed how sampling bias and data snooping inflate reported results [7]. We also found few systems that carry a detection through to a safe response and a governance record. Incident handling is normally left to other tools, although MITRE ATT&CK [16], ISO/IEC 27001:2022 [17] and NIST SP 800-30 [18] already give a common language for describing and ranking incidents. Our system attempts to cover the whole path from packet to audited firewall rule.

III. METHODOLOGY

A. System Architecture Overview
The system runs on Windows 10/11 under Python 3.11 and has six layers (Fig. 1). In the ingestion layer, Scapy [19] captures packets through the Npcap driver [20]; the same layer computes flow features, listens on decoy ports and reads the Windows Security log. The detection layer runs the ML models and the scan detector, uses SHAP to explain ML verdicts, and passes everything to a threat engine that correlates the sensors. Alerts, blocks and the audit log are stored in SQLite with write-ahead logging switched on, so the dashboard can read while sensors are writing. The remaining three layers handle firewall actions, incident mapping and scoring, and the Streamlit [21] operations conso

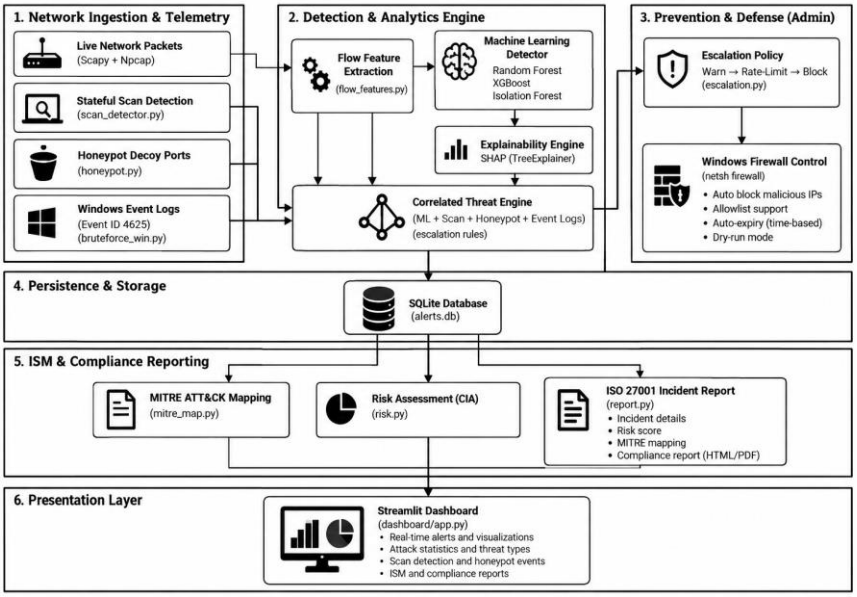


Fig. 1. System architecture of the AI-based NIDS/NIPS

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510117

B. Dataset and Leakage-Aware Preprocessing

We trained the ML core on CIC-IDS2017 [4] (state which daily CSV files were used and the number of flows per class). Shuffling the raw flows at random leaks information, because repeated probes and flood packets produce byte-identical records, and these end up in both splits. Table I lists the cleaning steps in the order we applied them. The RobustScaler was fitted on the training split only, and after splitting we confirmed that no sample appeared in both sets. Balanced class weights were used so that benign traffic would not dominate training.

TABLE I PREPROCESSING STEPS AND THEIR EFFECT

Step	Action	Effect
Identifier stripping	Drop Flow ID, IPs, ports, timestamp and duplicated CICFlowMeter columns	Removes host-specific shortcuts
Invalid value handling	Remove infinite and NaN values	162 non-finite cells removed
Deduplication	Drop exact duplicate flow records	12,537 flows removed (22.16%)
Constant pruning	Drop zero-variance features (e.g. Bwd PSH Flags)	8 features removed
Split and scale	Split ratio, stratified?, fit RobustScaler on training split only	No train/test overlap
Class weighting	Balanced class weights	Limits majority-class bias

C. Detection Models

We trained three models on the same features. Random Forest [11] is the baseline and XGBoost [12] the main supervised classifier. Isolation Forest [13] was fitted on benign flows only, so it measures how far a flow sits from normal behaviour without using attack labels. If a supervised model passes 99.5% test accuracy, a leakage check runs automatically. It reports the largest single-feature importance and the highest Pearson correlation between any feature and the label, and flags the model if either points to a shortcut. SHAP TreeExplainer [15] gives both a global ranking of features and a waterfall explanation for each individual alert.

D. Live Flow Feature Extraction

CICFlowMeter [9] is written in Java and computes its statistics once a connection has ended. That is fine for building a dataset but awkward for live capture. We wrote a bidirectional extractor in Python that works directly on Scapy packets and outputs the feature vector the trained models expect. Table II separates the features that match the original definitions from those we had to approximate.

TABLE II LIVE FLOW FEATURES AGAINST CICFLOWMETER

Category	Count	Features and method
Reproduced exactly	56	Flow duration, packet counts and lengths, packet length statistics, flow bytes/s and packets/s, flow/forward/backward IAT statistics, TCP flag counts, down/up ratio, segment sizes, initial window bytes
Approximated	12	Subflow packets and bytes modelled on connection state; active and idle statistics computed with a 5-second inter-packet gap threshold

E. Rule-Based Sensors

Per-flow learning is weak in some places, so three sensors fill the gaps. The **stateful scan detector** looks for flag combinations that break the TCP specification (Null, Xmas and FIN scans). It flags a fast SYN scan when one source reaches 15 or more distinct destination ports within a minute and its SYN-to-ACK ratio exceeds 3.0, and a slow scan when the rate stays under 2 packets per second across a ten-minute window. It also detects ICMP and ARP host sweeps.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510117

The **honeypot** opens listener sockets on ports 2222, 8080, 8443 and 33890. No real service runs on them, so any inbound connection produces a critical HONEYPOT_HIT alert. The **brute-force monitor** reads Windows Event ID 4625 (failed logon) and raises WIN_BRUTEFORCE after five failures from one source inside five minutes.

F. Prevention Engine and Safety Gates

Alerts go through the escalation policy in Table III, where the response grows with the number of offences from the same source. A honeypot hit skips straight to the top tier, since a legitimate client has no reason to connect to a decoy port.

TABLE III FOUR-TIER ESCALATION POLICY

Tier	Trigger	Action	Enforcement
1	First offence, low severity	WARN	Advisory entry in audit log
2	Second offence, medium severity	RATE_LIMIT	Active monitoring flag
3	Third offence, high severity	TEMP_BLOCK	Firewall block, 300 s
4	Fourth offence or honeypot hit	PERMANENT_BLOCK	Firewall block, 24 h or longer

Blocks are applied as netsh advfirewall rules named NIDS_BLOCK_<ip>. A request has to pass the gates shown in Fig. 3 before any rule is written. The engine will not block the default gateway, the DNS servers, the host's own adapter addresses, loopback or anything on the allowlist, and it ignores addresses outside the authorised subnets listed in the configuration file. By default it runs in dry-run mode, recording the rule without applying it. Every rule has an expiry time and a sweeper removes it when that time passes. If the program crashes, leftover NIDS_BLOCK rules are deleted on the next start. BLOCK, UNBLOCK, BLOCK_REJECTED and STARTUP_CLEANUP events all go into the audit log together with the reason and the raw command output.

G. Information Security Management Layer

Table IV shows how each alert type maps to a MITRE ATT&CK tactic and technique [16]. Risk is scored as

$$R = L \times C \quad (1)$$

Here L is a likelihood between 1.0 and 5.0, based on alert severity and adjusted by detection confidence, and C is the criticality of the destination asset on a 1 to 5 scale, taken from an asset inventory (a domain controller is 5, a database server 4, a workstation 2). Following the qualitative bands in NIST SP 800-30 [18], a score of 19 or above is Critical, 13 or above High, 6 or above Medium, and anything lower Low. The engine also records which CIA property is mainly at risk and which is secondarily affected. For each incident it can export a self-contained HTML report containing the incident ID, timestamps, source and destination, packet-level evidence, the related audit log entries and recommended ISO/IEC 27001:2022 Annex A controls [17], for example A.8.20 (network security), A.8.16 (monitoring activities), A.5.15 (access control), A.8.5 (secure authentication) and A.8.8 (management of technical vulnerabilities).

TABLE IV MAPPING OF DETECTIONS TO MITRE ATT&CK

Detection	Tactic	Technique
SYN, Null, Xmas, FIN and slow scans	Reconnaissance (TA0043)	T1595.001, T1046
ICMP and ARP host sweeps	Discovery (TA0007)	T1018
Event 4625 surge, ML brute force	Credential Access (TA0006)	T1110.001
Honeypot connection	Lateral Movement (TA0008)	T1021
ML DoS flood	Impact (TA0040)	T1498

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510117

Web attack; rogue device	Initial Access (TA0001)	T1190; T1200
Botnet beaconing	Command and Control (TA0011)	T1071
Unapproved open port	Persistence (TA0003)	T1571
Unsupervised anomaly	Defense Evasion (TA0005)	T1562

IV. RESULTS AND EVALUATION

A. Experimental Setup

All three models were trained and tested on the cleaned CIC-IDS2017 subset from Section III-B. We measured latency one flow at a time and report the median and the 95th percentile. Throughput was measured with batched prediction. The experiments ran on processor with RAM under Windows 10/11 edition and Python 3.11, without a GPU.

B. Detection Performance

Table V gives the results. XGBoost is ahead on every supervised metric. Its macro F1 is 1.85 points higher than that of Random Forest, so most of its advantage comes from the smaller attack classes, and its false positive rate is less than half. Isolation Forest, which never saw attack labels, has a false positive rate of 5.39%, about ten times that of XGBoost.

TABLE V DETECTION PERFORMANCE ON THE DEDUPLICATED TEST SPLIT

Metric	Random Forest	XGBoost	Isolation Forest
Accuracy	99.25%	99.62%	N/A
Macro F1-score	97.54%	99.39%	N/A
Weighted F1-score	99.25%	99.62%	N/A
False positive rate	1.22%	0.55%	5.39%
Median single-flow latency	35.55 ms	0.38 ms	2.82 ms
P95 single-flow latency	42.71 ms	0.73 ms	4.02 ms
Batch throughput (flows/s)	106,242	772,559	304,462

The larger difference between the models is speed. As Fig. 2 shows, XGBoost classifies one flow in a median of 0.38 ms, roughly 94 times faster than Random Forest, and handles 772,559 flows per second in batch mode. In live use each flow is scored as soon as it completes, so this gap matters far more than the 0.37-point difference in accuracy. We suspect the Random Forest figure is inflated by per-call overhead in its prediction path and not by evaluating the trees themselves; confirm: n_estimators / n_jobs setting would settle this.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510117

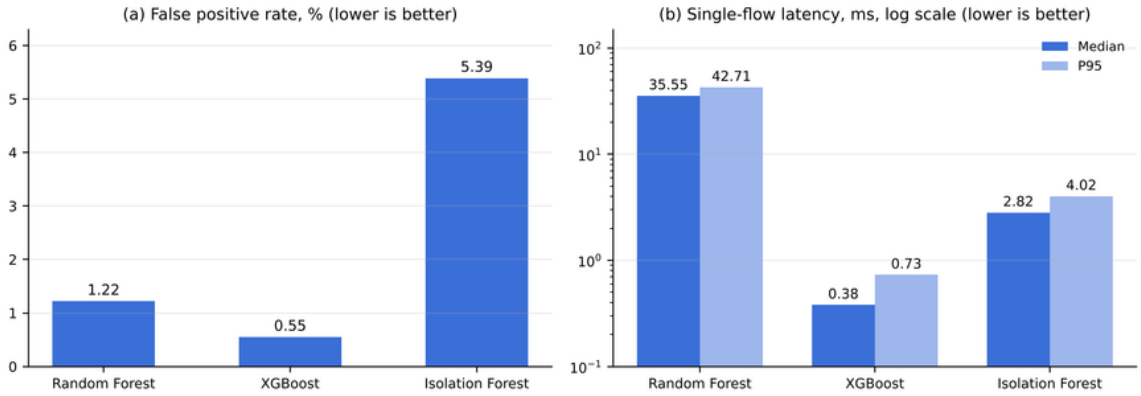


Fig. 2. False positive rate and single-flow latency per model

C. Zero-Day Held-Out Experiment

To see how the models cope with an attack they have never seen, we removed the whole PortScan family (324 flows after deduplication) from training. The supervised models can only pick from the classes they know, and they labelled more than 89% of these flows as benign. Isolation Forest recalled just 0.31%. The reason is that a scan probe is usually a single SYN packet, and its summary statistics look much like the start of a normal TCP handshake. We consider this the most useful negative result in the study. Per-flow features cannot describe reconnaissance, because a scan only shows up when many flows from one source are viewed together. This is what the stateful scan detector in Section III-E is for.

D. Leakage Verification and Explainability

Since XGBoost passed the 99.5% threshold, the leakage check ran. Bwd Packet Length Min was the most important feature but accounted for only 15.18% of total importance, and no feature had a correlation with the label above 0.95. SHAP ranked Bwd Packet Length Min, Fwd Packet Length Max, Bwd Packet Length Std and Bwd IAT Min as the strongest global drivers, and we generated waterfall plots for individual DoS, brute-force, web-attack and botnet alerts. These checks show that the model is not relying on one dominant feature. They cannot exclude all leakage, however. Flows belonging to the same attack session may still be split between training and test data, a point we return to in Section VI.

E. End-to-End Testbed Validation

We tested the detection and prevention path on a host-only virtual network, with the system running on a Windows defender and a Kali Linux machine as the attacker. Five scenarios were run: a fast SYN scan (nmap -sS) ending in a TEMP_BLOCK; Null, Xmas and FIN scans; a connection to decoy port 2222 ending in a PERMANENT_BLOCK; a manual unblock followed by timed expiry of a rule; and export of an ISO 27001 incident report, in which a critical alert against a database server scored 5.0 × 4.0 = 20.0 (Critical). Report the measured outcome of each scenario, e.g. alerts raised, time from first probe to block, number of probes before detection, over N repeated runs.

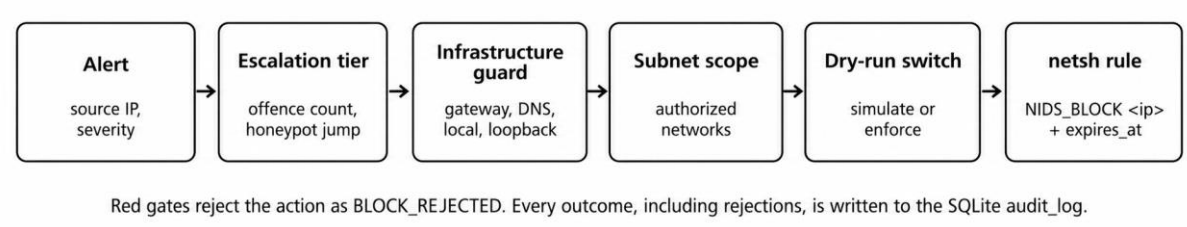


Fig. 3. Safety gates between an alert and a firewall rule

F. Failure Mode Analysis

We found four ways the system can fail. The first is the one the held-out test exposed: single-flow classifiers do not see reconnaissance. Second, an attacker can stay under the scan thresholds by probing fewer than 15 ports a minute while also keeping below the slow-scan window, or by spreading probes over many source addresses. Third, a spoofed source address could make the system block an innocent host. Subnet scoping, the infrastructure guards and rule expiry reduce this risk without eliminating it. Fourth, 12 of the 68 live features are approximations, so live vectors differ slightly from the training data, and any decision taken before a connection closes is based on incomplete statistics.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510117

V. SYSTEM IMPLEMENTATION AND DEPLOYMENT

A. Software Organisation

The code is split by phase into modules for ML (preprocessing, training, evaluation and SHAP), detection (pre-flight checks, capture, flow features, scan detection, honeypot and brute-force monitoring), prevention (blocking and escalation), scanning, ISM (MITRE mapping, risk and reports) and the dashboard. Thresholds and safety limits are kept in one configuration file. Before capture starts, pre-flight checks confirm that the process has administrator rights and that Npcap is installed. A separate proactive scanner, which only touches authorised subnets, discovers hosts through ARP, flags MAC addresses that are not in the asset baseline, lists open ports and service banners using Nmap (or plain sockets when Nmap is missing), reports changes since the last snapshot and looks up service versions against CVE records through the NIST NVD API [22]. There are 37 automated tests covering all six phases.

B. Operations Console and Audit Trail

The Streamlit console has five main views. The alert feed can be filtered by severity, type and IP and shows the raw JSON of each alert. The analytics view charts alerts by class and severity and lists the most active sources. The firewall view shows current and past rules with expiry countdowns and a button for manual unblocking. The scanner view holds the asset inventory and the CVE cache, and the ISM view previews an incident's MITRE mapping, risk score and CIA impact and lets the user download its HTML report. Actions taken by an administrator in the console are logged to the same audit table as automatic ones, so an incident can be followed from the first packet to the final state of the firewall.

VI. DISCUSSION

Our results suggest that machine learning and rules each have their place, and that the job is to give each the problems it handles well. XGBoost separates the attack classes it was trained on with a 0.55% false positive rate in under a millisecond per flow, which is fast enough for per-flow scoring on a single host. The held-out PortScan test shows the other side: over 89% of unseen scan flows were called benign. We chose to report that failure and design around it, using a stateful detector that looks at windows of packets and honeypot ports whose alerts do not depend on any trained model.

The headline accuracy should still be treated with caution. Removing duplicates and identifiers closes the most obvious leak, but CIC-IDS2017 has documented labelling problems [6] and was recorded in 2017, so it contains none of the current command-and-control frameworks or evasion methods. Our leakage check catches a single dominant feature or a very high feature-label correlation, but not leakage between related flows from the same session. A split by time or by session, plus testing on a second dataset, would give a more realistic estimate. Flow features also say nothing about encrypted payloads, so attacks hidden inside TLS sessions are beyond the reach of the system unless a terminating proxy is added.

On a production network automatic blocking is risky, and most of the prevention design follows from that. Enforcement is off by default, infrastructure addresses cannot be blocked, actions are limited to authorised subnets, every rule expires and every decision is logged. Mapping alerts to ATT&CK techniques and ISO 27001 controls means a security team can discuss an incident using the vocabulary of its own governance process instead of the internals of a model.

VII. CONCLUSION

We have described an AI-based network intrusion detection and prevention system for Windows hosts that combines leakage-aware machine learning with stateful rule-based sensors, guarded firewall enforcement and ISO/IEC 27001 incident reporting. On deduplicated CIC-IDS2017 flows, XGBoost achieved 99.62% accuracy, 99.39% macro F1 and a 0.55% false positive rate at a median of 0.38 ms per flow. Holding out the PortScan family showed that per-flow models miss reconnaissance they have not seen before, and the stateful scan detector was added to cover that case. Prevention is kept safe through a four-tier escalation policy, infrastructure guards, subnet scoping, rule expiry and a dry-run default, and MITRE ATT&CK mapping together with quantitative risk scoring makes each incident auditable.

In future work we plan to evaluate the models with session-grouped and time-based splits and on a second dataset, to measure the scan detector and the full detection-to-block path on the testbed, to try sequence-aware models for attacks that span many flows, and to replace the fixed scan thresholds with adaptive ones.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026
DOI: 10.17148/IJARCCE.2026.1510117

REFERENCES

[1]. D. E. Denning, *An intrusion-detection model,* IEEE Trans. Software Engineering, vol. SE-13, no. 2, pp. 222-232, 1987.
[2]. M. Roesch, *Snort: Lightweight intrusion detection for networks,* Proc. USENIX LISA, pp. 229-238, 1999.
[3]. V. Paxson, *Bro: A system for detecting network intruders in real-time,* Computer Networks, vol. 31, no. 23-24, pp. 2435-2463, 1999.
[4]. I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, *Toward generating a new intrusion detection dataset and intrusion traffic characterization,* Proc. ICISSP, pp. 108-116, 2018.
[5]. R. Sommer and V. Paxson, *Outside the closed world: On using machine learning for network intrusion detection,* Proc. IEEE Symposium on Security and Privacy, pp. 305-316, 2010.
[6]. G. Engelen, V. Rimmer, and W. Joosen, *Troubleshooting an intrusion detection dataset: The CICIDS2017 case study,* Proc. IEEE Security and Privacy Workshops, pp. 7-12, 2021.
[7]. D. Arp et al., *Dos and don'ts of machine learning in computer security,* Proc. USENIX Security Symposium, 2022.
[8]. J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan, *Fast portscan detection using sequential hypothesis testing,* Proc. IEEE Symposium on Security and Privacy, pp. 211-225, 2004.
[9]. A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, *Characterization of Tor traffic using time based features,* Proc. ICISSP, pp. 253-262, 2017.
[10]. M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, *A survey of network-based intrusion detection data sets,* Computers & Security, vol. 86, pp. 147-167, 2019.
[11]. L. Breiman, *Random forests,* Machine Learning, vol. 45, no. 1, pp. 5-32, 2001.
[12]. T. Chen and C. Guestrin, *XGBoost: A scalable tree boosting system,* Proc. ACM SIGKDD, pp. 785-794, 2016.
[13]. F. T. Liu, K. M. Ting, and Z.-H. Zhou, *Isolation forest,* Proc. IEEE ICDM, pp. 413-422, 2008.
[14]. S. M. Lundberg and S.-I. Lee, *A unified approach to interpreting model predictions,* Advances in Neural Information Processing Systems 30, pp. 4765-4774, 2017.
[15]. S. M. Lundberg et al., *From local explanations to global understanding with explainable AI for trees,* Nature Machine Intelligence, vol. 2, no. 1, pp. 56-67, 2020.
[16]. B. E. Strom et al., *MITRE ATT&CK: Design and philosophy,* The MITRE Corporation, 2018. [Online]. Available: https://attack.mitre.org/
[17]. ISO/IEC 27001:2022, *Information security, cybersecurity and privacy protection - Information security management systems - Requirements,* ISO, Geneva, 2022.
[18]. NIST, *Guide for conducting risk assessments,* Special Publication 800-30 Rev. 1, 2012.
[19]. P. Biondi and the Scapy community, *Scapy documentation.* [Online]. Available: https://scapy.readthedocs.io/
[20]. Nmap Project, *Npcap: Windows packet capture library and driver.* [Online]. Available: https://npcap.com/
[21]. Streamlit Inc., *Streamlit documentation.* [Online]. Available: https://docs.streamlit.io/
[22]. NIST, *National Vulnerability Database API 2.0 documentation.* [Online]. Available: https://nvd.nist.gov/developers

 This work is licensed under a Creative Commons Attribution 4.0 International License