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.2026.1510113

SentinelGuard: Explainable Detection of Spyware and Remote Access Trojans on Windows

Nirbhay A. Shende[1*], Vishal V. Ghorad[1], Soumya P. Mendse[1], Kanak V. Gupta[1], Radhika P. Malu[2]

Student, Dept. of Artificial Intelligence and Data Science, Prof. Ram Meghe Institute of Technology & Research, Badnera, Amravati, India[1]

Professor, Dept. of Artificial Intelligence and Data Science, Prof. Ram Meghe Institute of Technology & Research, Badnera, Amravati, India[2]

*Corresponding Author

Abstract: Remote access trojans (RATs) and other spyware can secretly use a computer's camera, contact command-and-control (C2) servers and upload data, while signature-based protection may miss new variants. This paper presents SentinelGuard, a software-only monitor for Windows personal computers that learns the normal behaviour of the PC on which it is installed and combines an unsupervised anomaly detector with exact rules. A per-feature normal-range model and an Isolation Forest operate on seven behavioural features, and SHAP values explain every AI alert in plain language; rules cover camera or microphone use by unexpected applications, per-application C2-style connection bursts, unknown local-network devices and strong unknown WiFi networks. To address RAT delivery, a Downloads guard inspects the Windows Mark-of-the-Web of new files and an optional, read-only email monitor checks attachments and links. A self-protecting learner removes rare upload bursts before training and retrains on a rolling 14-day window. On 215 held-out normal readings the false-alarm rate was 1.4%, and five synthetic attack scenarios were all flagged. In live, safe simulations the AI model detected cryptominer-like CPU load (HIGH) and data upload (CRITICAL) with explanations naming the responsible feature, the C2 rule detected every simulated connection burst, a second PC detected all three demonstration steps including camera use, and test RAT-delivery downloads and emails were flagged within 7 s and 13 s. Self-cleaning lowered a learned upload limit inflated by a burst from about 4,201 KB/s to about 298 KB/s. Camera attribution failed on an updated Windows 11 build, and all attacks were simulated.

Keywords: Anomaly detection, explainable artificial intelligence, Isolation Forest, remote access trojan, SHAP, spyware detection, Windows security.

I. INTRODUCTION

Remote access trojans give an attacker interactive control over a victim's computer. Valeros and Garcia documented how RATs grew and became commoditised over three decades, with ready-made tools sold or shared [1], and Farinholt *et al.* observed operators of the DarkComet RAT frequently attempting to access victims' webcams [2]. Signature-based detection identifies known malware families reliably but is weak against obfuscated and previously unseen samples [3], and zero-day attacks may remain undisclosed for long periods [4]. RATs commonly reach a computer through email, as a malicious attachment or a link that leads to malware [5], [6].

Behavioural detection on the user's own computer is a natural complement, but it must satisfy constraints that research systems often do not: it must run without administrator rights or kernel components, learn what is normal for one particular PC, keep false alarms low enough for a home user, and explain each alert in terms that a non-expert can act upon.

This paper presents SentinelGuard, a prototype that addresses these constraints, and reports its results. The contributions are:

- a per-PC hybrid detection pipeline that combines a learned normal-range model and an Isolation Forest over seven behavioural features with exact rules for device use, per-application C2-style bursts, unknown local hosts and strong unknown WiFi networks;
- a severity policy based on the kind of signal, an alert policy that suppresses floods, and plain-language explanations backed by SHAP values;

 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.2026.1510113

- protection at the point of RAT delivery: a Downloads guard based on the Windows Mark-of-the-Web and an optional read-only email monitor;
- a self-protecting learner that removes rare upload bursts before training and keeps the model current without user maintenance; and
- an evaluation with held-out normal data, synthetic scenarios, live safe simulations on two PCs, and email and download tests.

Section II reviews related work, Section III describes the design, Section IV the implementation, Section V the experimental setup, Section VI the results, Section VII the discussion and limitations, and Section VIII concludes.

II. RELATED WORK

A. *Behavioural and Anomaly-Based Detection*

Dynamic malware analysis observes what a program does, such as system calls, file and registry activity and network use, but the surveyed systems are mainly analysis environments for researchers [7]. Forrest *et al.* defined normal behaviour as the short system-call sequences of a process and treated deviations as intrusions [8], establishing detection relative to the normal behaviour of a specific system. Surveys of anomaly detection [9] and of network and intrusion detection [10], [11], [12] describe the main families of methods and note that unsupervised methods are useful when labelled attacks are unavailable, while false alarms remain a central problem. Sommer and Paxson explained why machine learning is difficult to apply to network intrusion detection, citing the high cost of errors, the semantic gap between an anomaly and an actionable alert, and the difficulty of evaluation [13]. Benchmark datasets such as UNSW-NB15 [14] support comparison of methods but describe laboratory network traffic rather than the behaviour of one personal computer.

B. *Isolation Forest*

Isolation Forest isolates observations by random recursive partitioning; anomalies need fewer splits and therefore have shorter average path lengths [15], and the extended analysis appears in [16]. The method has linear time complexity and low memory use, but it does not by itself state which feature caused an anomaly. SentinelGuard uses the scikit-learn implementation [17].

C. *RAT and Command-and-Control Detection*

Gardiner and Nagaraja surveyed machine-learning approaches to C&C detection and their vulnerability to adaptive attackers [18]. Anderson and McGrew classified encrypted malware traffic from flow metadata and studied noisy labels and non-stationarity [19]. Such flow-level methods require packet capture and labelled traffic, which a user-level monitor on a home PC does not have.

D. *Explainable AI for Anomaly Detection*

SHAP assigns each feature a contribution to a prediction based on Shapley values [20], and TreeExplainer computes exact SHAP values for tree ensembles in polynomial time [21]. LIME explains individual predictions with a local surrogate model [22]. Amarasinghe *et al.* argued that anomaly detectors in cyber-security must explain their detections to operators [23].

E. *Research Gap*

The building blocks above are mostly studied separately, evaluated on research datasets, or designed for organisational security teams. Based on the literature reviewed, the authors did not find a free tool for individual Windows users that combines per-PC learning, rules for device use and C2-style bursts, explanations for non-experts, and checks at the point of RAT delivery. Table I summarises the positioning qualitatively.

TABLE I QUALITATIVE COMPARISON OF APPROACHES

Aspect	Signature antivirus	OS privacy indicators	SentinelGuard
Detects unknown behaviour	Limited [3]	No (only shows use)	Yes (anomaly model + rules)
Camera / microphone use	Not main focus	Icon and app list; no alert	Alert naming the app
Explains alerts	Threat name	Not applicable	SHAP + plain-language reason
Learns this PC	No	No	Yes
Removes / blocks threats	Yes	No	No (alert only)
Covers all malware types	Yes (known signatures)	No	No (surveillance behaviour only)

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.2026.1510113

III. SYSTEM DESIGN

A. *Architecture*

Fig. 1 shows the architecture. Five monitor threads collect data independently: process and resource metrics every 5 s, the network connection table every 5 s, nearby WiFi networks every 30 s, new files in the Downloads folder every 3 s and, if the user connects a mailbox, new emails every 60 s. Each result is stored with its collection time in a shared state. A detection loop runs every 5 s, discards data older than three monitor intervals, passes the latest readings to the AI detector and the rule engine, and hands all detections to an alert manager, which stores incidents and raises Windows notifications. A learner receives the normal readings of each cycle and keeps the model current.

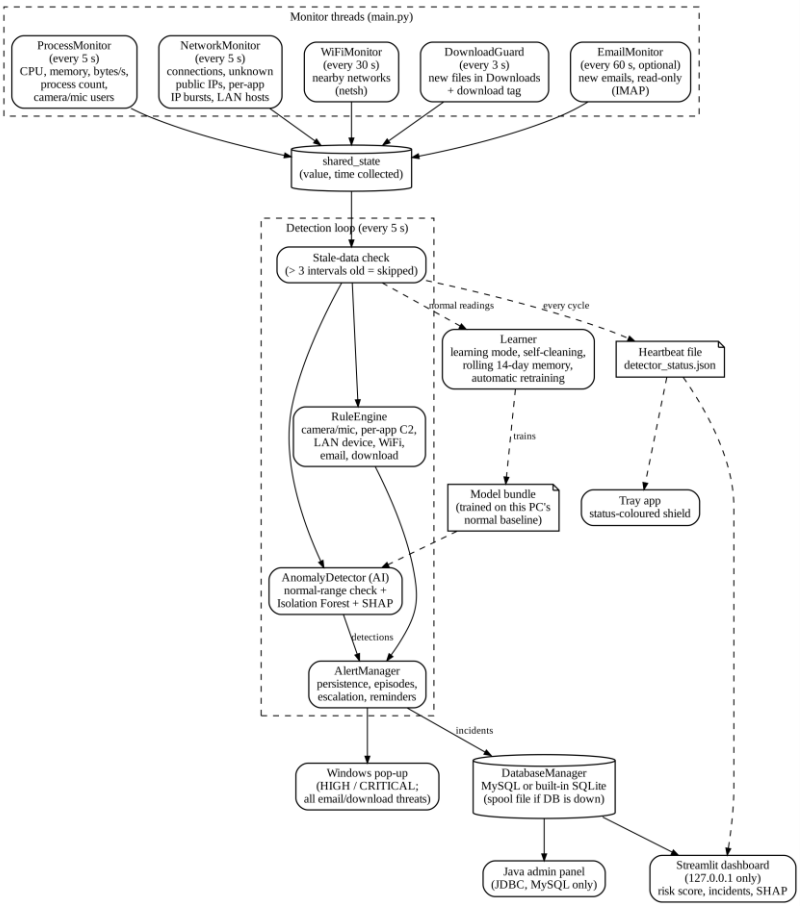


Fig. 1 Architecture of SentinelGuard: monitor threads, detection loop, learner, storage and user interfaces.

B. *Features*

The AI model uses seven features (Table II). *Surveillance* features describe data leaving the PC or contact with hosts; *context* features describe general load. The distinction drives the severity policy.

TABLE II FEATURES USED BY THE AI MODEL

Feature	Meaning	Group
cpu_percent	Total CPU utilisation (%)	Context
memory_percent	Memory utilisation (%)	Context
bytes_sent_per_sec	Outbound data rate	Surveillance
bytes_recv_per_sec	Inbound data rate	Context
active_connections	Active network connections	Surveillance
running_processes	Running processes	Context
unknown_ip_connections	Distinct untrusted public IPs (60 s window)	Surveillance

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.2026.1510113

C. *Normal-Range Model and Isolation Forest*

For each feature *j*, training computes the minimum, maximum, median (p50) and 99th percentile (p99) of the normal readings. With span $s_j = \max(\max_j - \min_j, 1)$ and spread $d_j = \max(p99_j - p50_j, 1)$, the normal limit E_j and the critical limit C_j are

$$E_j = \max(p99_j + 1, \min(\max_j + 0.25s_j, p99_j + 2d_j)) \quad (1)$$

$$C_j = \max(E_j, \min(\max_j + s_j, p99_j + 5d_j)) \quad (2)$$

Taking the smaller of a maximum-based and a percentile-based estimate prevents one extreme baseline reading from making a limit unreachable; the CPU critical limit is capped at 95%. An Isolation Forest with 300 trees and contamination 0.01 is trained on the same readings to detect unusual *combinations* of values that are individually within range, and the lowest training decision score is kept as a reference. At least 200 readings are required. For every anomalous reading, TreeExplainer [21] computes feature contributions, and the description names the responsible feature, its value and the normal reference.

D. *Severity and Alert Policy*

Severity depends on which kind of feature is out of range, not only on how unusual a reading is (Table III), so that a busy PC cannot by itself produce a CRITICAL alert. AI detections must persist in two of the last three cycles (three of three for LOW); rule detections are raised immediately. Each situation has an episode key, so a continuing situation yields one incident, with a new incident on escalation and a reminder every 15 min.

TABLE III SEVERITY POLICY FOR AI DETECTIONS

Condition	Severity
Isolation Forest only (score < 0)	LOW
Isolation Forest only, below lowest training score	MEDIUM
Context feature(s) above normal limit only	MEDIUM
Context feature above critical limit	HIGH
One surveillance feature above normal limit	HIGH
Surveillance hit plus a second hit, or above critical limit	CRITICAL

E. *Detection Rules*

Signals that are precise enough need no model (Table IV). The camera and microphone rule reads the Windows privacy records that name the application using the device. The C2 rule counts, per application, the distinct public servers contacted in a 60 s window, using only the operating system's connection table.

TABLE IV DETECTION RULES

Rule	Condition	Severity
Camera / microphone	Application not in the allowed list uses the device	HIGH
C2-style burst	One application contacts ≥ 20 distinct public servers within 60 s	HIGH
Unknown LAN device	Untrusted private-network host has an inbound connection	HIGH
Strong unknown WiFi	Untrusted network visible with signal ≥ 80%	MEDIUM
Email threat	Dangerous or disguised attachment, program inside a ZIP, or link that downloads a program	HIGH / MEDIUM
Dangerous download	Program, macro document or ZIP with a program from an email site, bare IP or look-alike site; or disguised name	HIGH / MEDIUM

F. *Protection at the Point of RAT Delivery*

When a browser saves a file, Windows records its origin in a hidden NTFS alternate data stream named Zone.Identifier, the Mark-of-the-Web [24], which can contain the download address (HostUrl) and the referring page (ReferrerUrl) [25]. The Downloads guard lists the Downloads folder every 3 s, waits until a new file is complete, reads this stream and checks the file name. A file is flagged if its name is disguised, for example by a decoy double extension or by a right-to-left override character [26], or if it is a program, script, disk image or macro document, or a ZIP file containing one, that came from an email website, a bare IP address or a punycode [27] look-alike domain. Ordinary installers from ordinary websites are not flagged.

The optional email monitor connects over IMAP [28] with an app password, selects the inbox read-only and fetches new messages with BODY.PEEK so that nothing is marked as read, moved or deleted. Attachments are checked by name and

 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.2026.1510113

links by their address and visible text. Neither component opens, runs or downloads anything. Email and download threats always raise a notification, also at MEDIUM severity, because the warning is only useful before the file is opened.

G. *Self-Protecting Learning*

A model trained once on a short baseline can be distorted by a single event during learning, such as a large upload, and becomes outdated as habits change. Before every training the learner therefore removes outbound-rate readings above

$$\tau = \mathrm{p95} + K\cdot(\mathrm{p95} - \mathrm{p50}),\ \ K = 10 \tag{3}$$

but only if at most 5% of the readings exceed τ. Readings are not recorded while a rule fires, while a HIGH or CRITICAL AI alert is active, or while a demonstration runs. After the first model, the learner retrains 24 h later and then every 7 days on the readings of the last 14 days, swapping in a new detector without pausing detection; if retraining fails, the current model is kept.

IV. IMPLEMENTATION

SentinelGuard is written in Python. psutil [29] supplies process, CPU, memory, network and connection information; scikit-learn [17] the Isolation Forest; the SHAP library TreeExplainer [21]; and Streamlit [30] a local dashboard that is reachable only from the same PC. Incidents are stored in SQLite by default or in MySQL when configured, with a spool file that preserves incidents while the database is unavailable. A tray application shows the protection state and raises Windows notifications, and a Java Swing panel offers incident review on MySQL. A per-user installer bundles an embeddable Python 3.14.2 and needs no administrator rights. The automated test suite (pytest) passed 410 tests with 12 skipped in the Linux development environment; skipped tests require MySQL or Windows. The source code and installer are publicly available at: https://github.com/nirbhayninja/SentinelGuard.

V. EXPERIMENTAL SETUP

Table V lists the environments. All attacks were safe simulations written by the authors; no malware was used.

TABLE V TEST ENVIRONMENTS

Item	Details
Development PC	ASUS laptop, Windows 11 Home Single Language (version 26H2, OS build 26300.9711), Intel Core i7-11800H @ 2.30 GHz, 16 GB RAM
Second PC	Windows 11 Home Single Language, version 25H2 (OS Build 26200.9711)
Baseline	1,074 normal readings (about 89 min at 5 s intervals)
Email account	Separate Gmail test account, read-only access with an app password
Network	Wi-Fi and mobile hotspot connections

A. *Offline Evaluation*

The baseline was split in time order: a temporary model was trained on the oldest 80% (859 readings) and tested on the newest 20% (215 readings). The false-alarm rate is the share of held-out normal readings flagged. False incidents per hour were obtained by replaying the held-out readings through the alert manager. For five synthetic scenarios, only the features that the attack would affect were changed in every held-out reading, with magnitudes relative to the PC's own 99th percentile.

B. *Live Safe Simulations*

A cryptominer-like scenario loaded all CPU cores to 100% for 30 s; a data-upload scenario uploaded random data to a public speed-test server for 30 s with parallel streams; and a RAT-like scenario opened the camera for about 15 s while opening about 45 HTTPS connections to well-known public websites. The scenarios were run on the development laptop on 12–14 October 2026 and on a second PC on 15 October 2026. The package installed on the second PC contained the model trained on the development laptop, as shown by identical normal-maximum values in its alert descriptions.

C. *Email, Download and Self-Cleaning Tests*

The download test saved a harmless text file named Invoice_October.pdf.docm in the Downloads folder with the Zone.Identifier content that a browser writes for a Gmail attachment and removed it after 15 s. The email test sent a harmless message from the connected test account to itself, with an attachment named like a disguised macro document

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.2026.1510113

and a link to a program on a documentation-only IP address. Self-cleaning was checked on the development laptop's real baseline, which contained a short upload of about 3 MB/s recorded during learning.

VI. RESULTS

A. *Offline Evaluation*

Table VI gives the results. Three of 215 held-out normal readings were flagged (1.4%), and replay through the alert manager produced one MEDIUM incident in about 18 min of data, which extrapolates to 3.35 per hour; this figure rests on a single incident and is uncertain. All five synthetic scenarios were flagged in every reading. The stealth-RAT scenario was detected mostly at LOW severity and would not raise a notification.

TABLE VI RESULTS OF THE OFFLINE EVALUATION

Measure / scenario	Synthetic change	Result
False-alarm rate	Held-out normal readings	1.4% (3 of 215)
False incidents	Replay through alert manager	3.35 per hour
Data exfiltration	Upload 5× p99	100% (214 HIGH, 1 CRITICAL)
C2 burst	Unknown IPs 1.5×, connections 1.3× p99	100% (215 CRITICAL)
Cryptominer	CPU at 97%	100% (215 HIGH)
Process storm	40 processes above p99	100% (215 MEDIUM)
Stealth RAT	CPU 40%, upload 2×, unknown IPs 1.1× p99	100% (213 LOW, 2 MEDIUM)

B. *Live Safe Simulations*

Table VII lists the live results. On the development laptop the AI model detected the cryptominer-like load as HIGH PROCESS_ANOMALY and the upload as CRITICAL NETWORK_ANOMALY in every run, and the C2 rule detected the burst of 44–45 servers in both RAT-like runs. The camera rule detected the camera on 12 October but not on 14 October (Section VII). On the second PC all three steps were detected, including the camera step, with detection delays of 10–11 s for the AI scenarios. Fig. 2 shows the SHAP explanation of a CRITICAL upload detection, in which the outbound data rate is the largest contributor.

TABLE VII RESULTS OF THE LIVE SAFE SIMULATIONS

Scenario (date)	Observed incident(s)	Result
Cryptominer (13 Oct)	HIGH PROCESS_ANOMALY, CPU 100% (normal max 61.6%)	Detected
Upload (13 Oct)	CRITICAL NETWORK_ANOMALY, 4.8 MB/s (normal max 582.2 KB/s)	Detected
RAT-like (12 Oct)	CAMERA_ACCESS HIGH; C2_COMMUNICATION HIGH (44 servers)	Detected
RAT-like (14 Oct)	C2_COMMUNICATION HIGH (45 servers); no camera incident	Partial
Cryptominer (14 Oct)	HIGH PROCESS_ANOMALY, CPU 100%	Detected
Upload (14 Oct)	CRITICAL NETWORK_ANOMALY, 3.3 MB/s	Detected
RAT-like, PC 2 (15 Oct)	CAMERA_ACCESS HIGH; C2_COMMUNICATION HIGH (45 servers) 5 s later	Detected
Cryptominer, PC 2 (15 Oct)	HIGH PROCESS_ANOMALY, CPU 100%, after 11 s	Detected
Upload, PC 2 (15 Oct)	CRITICAL NETWORK_ANOMALY, 1.3 MB/s, after 10 s (44 MB at 1,193 KB/s)	Detected

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.2026.1510113

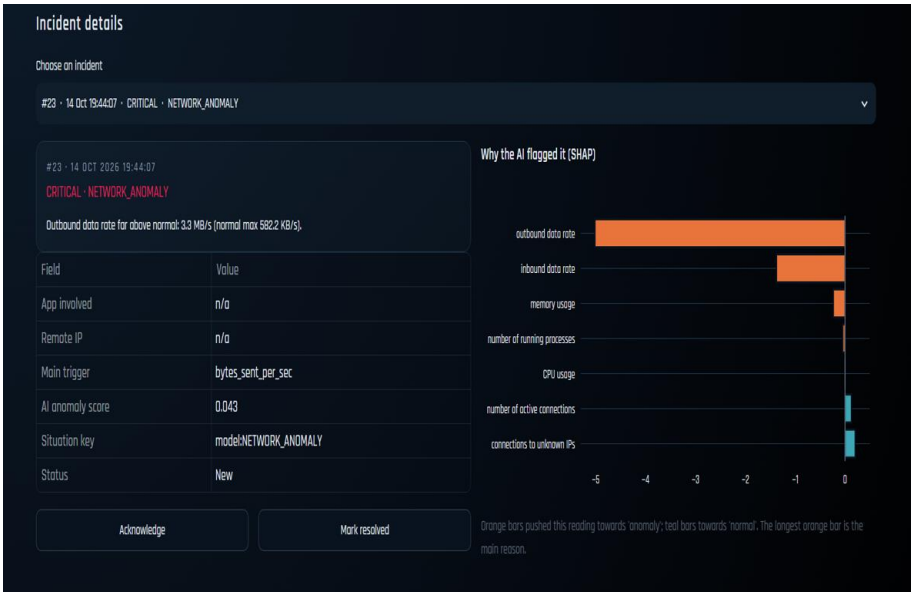


Fig. 2 Incident details of a CRITICAL NETWORK_ANOMALY (14 October 2026) with its SHAP explanation; the outbound data rate is the main contributor.

C. *Email and Download Threats*

Both RAT-delivery tests were detected (Table VIII). The download was flagged as HIGH DOWNLOAD_THREAT 7 s after the file was saved, with the reasons that it was downloaded from an email website and that a file pretending to be a PDF was a macro document (Fig. 3). The test email was flagged as HIGH EMAIL_THREAT 13 s after it was sent, before it was opened. Sixty-one automated tests of the email rules, the email monitor, the connection command and the Downloads guard passed, including tests that ordinary documents, images and installers from ordinary websites are not flagged.

TABLE VIII EMAIL AND DOWNLOAD THREAT TESTS (23 OCTOBER 2026)

Test	Observed	Result
Disguised attachment downloaded	HIGH DOWNLOAD_THREAT and notification after 7 s	Detected
RAT-delivery test email	HIGH EMAIL_THREAT and notification after 13 s	Detected
Ordinary emails and files	No alert (automated tests)	As expected

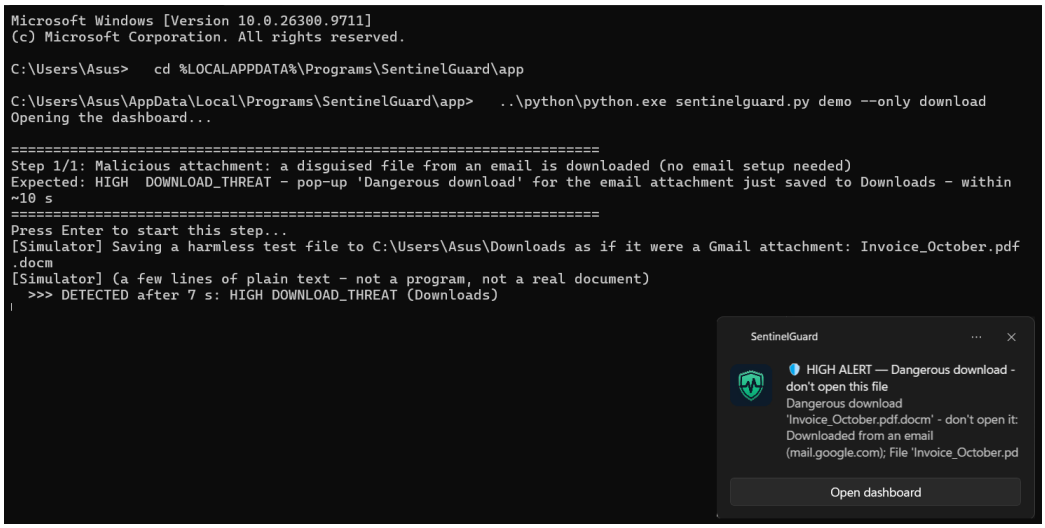


Fig. 3 Download test: the demonstration step reports a HIGH DOWNLOAD_THREAT after 7 s, and the Windows notification names the reasons.

 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.2026.1510113

D. *Self-Protecting Learning*

Without cleaning, the short burst in the real baseline raised the learned outbound limit to about 4,201 KB/s, so that simulated uploads of 1.2–3.3 MB/s stayed below it. With cleaning, the burst readings were removed and the limit returned to about 298 KB/s, at which the same uploads were reported as CRITICAL, while the false-alarm rate on held-out normal readings was the same in both cases (Table IX).

TABLE IX EFFECT OF SELF-CLEANING ON THE BASELINE

Quantity	Without cleaning	With cleaning
Learned outbound limit	≈ 4,201 KB/s	≈ 298 KB/s
Uploads of 1.2–3.3 MB/s	Below limit	CRITICAL
Held-out false-alarm rate	Same	Same

E. *False Alarms in Normal Use*

During about two hours of normal use on 14 October 2026, eight incidents were recorded without any simulation: five MEDIUM, two LOW and one HIGH, caused by CPU, memory and inbound-data levels that had not occurred during the 89-minute baseline. The severity policy kept most of them at MEDIUM or LOW because these are context features; one large download escalated to HIGH by exceeding the critical limit.

VII. DISCUSSION AND LIMITATIONS

The results indicate that per-PC, explainable anomaly detection combined with exact rules is practical on a home Windows PC without administrator rights: the AI model detected the CPU and upload scenarios with explanations that named the correct feature, the C2 rule detected every simulated burst, and the delivery-stage checks flagged both RAT-delivery tests before the file or email was opened. The severity policy limited the consequences of the short baseline, since most false alarms stayed at MEDIUM or LOW. The following limitations apply.

- **Camera attribution.** On the development laptop after a Windows 11 update (OS build 26300), desktop-application camera history was no longer available to a user-level program, so the camera rule did not fire on 14 October, although it worked on 12 October and on the second PC (OS build 26200).
- **Short baseline.** The model was trained on about 89 minutes of readings, which caused false alarms during heavy legitimate use; self-cleaning and rolling retraining are intended to reduce this over time but were evaluated only on one baseline.
- **Second PC.** The package on the second PC contained the development laptop's model, so the AI results there test the portability of the pipeline rather than learning on that PC.
- **Simulations only.** No real malware was used; the results show that the detection logic responds to the simulated behaviours but do not establish detection rates for real attacks, and an attacker could keep activity within the learned limits.
- **Scope of delivery checks.** The email check needs an app password, and a file downloaded through an email link from an ordinary website is flagged only when its name is disguised or its source is a bare IP address or look-alike domain.
- **Heuristic thresholds and metadata only.** The C2 and WiFi thresholds are heuristics, and without packet capture or kernel components the system cannot inspect traffic contents.

VIII. CONCLUSION AND FUTURE WORK

SentinelGuard combines a per-PC normal-range model and an Isolation Forest with SHAP explanations, exact rules for device use and C2-style bursts, checks at the point of RAT delivery, and a self-protecting learner in a per-user Windows application. On held-out normal data the false-alarm rate was 1.4%, all synthetic scenarios were flagged, and live safe simulations on two PCs and RAT-delivery tests were detected with explanations or reasons. Future work includes camera and microphone attribution on new Windows builds through a consented system service, evaluation on more PCs over longer periods and against real-world samples in a controlled environment, user feedback to mark false alarms as normal, sign-in with the mail provider instead of an app password, and code signing of the installer.

ACKNOWLEDGMENT

The authors thank **Dr. S. R. Gupta**, Head of the Department of Artificial Intelligence and Data Science, Prof. Ram Meghe Institute of Technology & Research, for support. This work received no external funding.

DOI: 10.17148/IJARCCE.2026.1510113

REFERENCES

[1]. V. Valeros and S. Garcia, "Growth and commoditization of remote access trojans," in *Proc. IEEE Eur. Symp. Security Privacy Workshops (EuroS&PW)*, 2020, pp. 454–462.

[2]. B. Farinholt, M. Rezaeirad, P. Pearce, H. Dharmdasani, H. Yin, S. Le Blond, D. McCoy, and K. Levchenko, "To catch a ratter: Monitoring the behavior of amateur DarkComet RAT operators in the wild," in *Proc. IEEE Symp. Security Privacy (SP)*, San Jose, CA, USA, 2017, pp. 770–787, doi: 10.1109/SP.2017.48.

[3]. E. Gandotra, D. Bansal, and S. Sofat, "Malware analysis and classification: A survey," *J. Inf. Security*, vol. 5, no. 2, pp. 56–64, 2014, doi: 10.4236/jis.2014.52006.

[4]. L. Bilge and T. Dumitraş, "Before we knew it: An empirical study of zero-day attacks in the real world," in *Proc. ACM Conf. Comput. Commun. Security (CCS)*, Raleigh, NC, USA, 2012, pp. 833–844, doi: 10.1145/2382196.2382284.

[5]. The MITRE Corporation, "Phishing: Spearphishing Attachment (T1566.001)," MITRE ATT&CK, v2.2. [Online]. Available: https://attack.mitre.org/techniques/T1566/001/ (accessed Oct. 26, 2026).

[6]. The MITRE Corporation, "Phishing: Spearphishing Link (T1566.002)," MITRE ATT&CK, v2.8. [Online]. Available: https://attack.mitre.org/techniques/T1566/002/ (accessed Oct. 26, 2026).

[7]. M. Egele, T. Scholte, E. Kirda, and C. Kruegel, "A survey on automated dynamic malware-analysis techniques and tools," *ACM Comput. Surv.*, vol. 44, no. 2, Art. no. 6, 2012, doi: 10.1145/2089125.2089126.

[8]. S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for Unix processes," in *Proc. IEEE Symp. Security Privacy*, Oakland, CA, USA, 1996, pp. 120–128, doi: 10.1109/SECPRI.1996.502675.

[9]. V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, Art. no. 15, 2009, doi: 10.1145/1541880.1541882.

[10]. M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 303–336, 2014, doi: 10.1109/SURV.2013.052213.00046.

[11]. H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 16–24, Jan. 2013, doi: 10.1016/j.jnca.2012.09.004.

[12]. A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2016, doi: 10.1109/COMST.2015.2494502.

[13]. R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proc. IEEE Symp. Security Privacy (SP)*, Oakland, CA, USA, 2010, pp. 305–316, doi: 10.1109/SP.2010.25.

[14]. N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Military Commun. Inf. Syst. Conf. (MilCIS)*, Canberra, ACT, Australia, 2015, doi: 10.1109/MilCIS.2015.7348942.

[15]. F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. 8th IEEE Int. Conf. Data Mining (ICDM)*, Pisa, Italy, 2008, pp. 413–422, doi: 10.1109/ICDM.2008.17.

[16]. F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Trans. Knowl. Discovery Data*, vol. 6, no. 1, Art. no. 3, 2012, doi: 10.1145/2133360.2133363.

[17]. F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[18]. J. Gardiner and S. Nagaraja, "On the security of machine learning in malware C&C detection: A survey," *ACM Comput. Surv.*, vol. 49, no. 3, Art. no. 59, 2016, doi: 10.1145/3003816.

[19]. B. Anderson and D. McGrew, "Machine learning for encrypted malware traffic classification: Accounting for noisy labels and non-stationarity," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, Halifax, NS, Canada, 2017, pp. 1723–1732, doi: 10.1145/3097983.3098163.

[20]. S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 30, Long Beach, CA, USA, 2017, pp. 4765–4774.

[21]. S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee, "From local explanations to global understanding with explainable AI for trees," *Nature Mach. Intell.*, vol. 2, no. 1, pp. 56–67, Jan. 2020, doi: 10.1038/s42256-019-0138-9.

[22]. M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?': Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, San Francisco, CA, USA, 2016, pp. 1135–1144, doi: 10.1145/2939672.2939778.

[23]. K. Amarasinghe, K. Kenney, and M. Manic, "Toward explainable deep neural network based anomaly detection," in *Proc. 11th Int. Conf. Human System Interaction (HSI)*, Gdańsk, Poland, 2018, pp. 311–317, doi: 10.1109/HSI.2018.8431364.



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.2026.1510113

[24]. The MITRE Corporation, "Subvert Trust Controls: Mark-of-the-Web Bypass (T1553.005)," MITRE ATT&CK, v2.0. [Online]. Available: https://attack.mitre.org/techniques/T1553/005/ (accessed Oct. 26, 2026).
[25]. C. Wilson, "Forensic analysis of the Zone.Identifier stream," Digital Detective, Oct. 8, 2021. [Online]. Available: https://www.digital-detective.net/forensic-analysis-of-zone-identifier-stream/ (accessed Oct. 26, 2026).
[26]. The MITRE Corporation, "Masquerading: Right-to-Left Override (T1036.002)," MITRE ATT&CK, v2.0. [Online]. Available: https://attack.mitre.org/techniques/T1036/002/ (accessed Oct. 26, 2026).
[27]. A. Costello, "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)," RFC 3492, Internet Engineering Task Force, Mar. 2003, doi: 10.17487/RFC3492.
[28]. A. Melnikov and B. Leiba, "Internet Message Access Protocol (IMAP) – Version 4rev2," RFC 9051, Internet Engineering Task Force, Aug. 2021, doi: 10.17487/RFC9051.
[29]. G. Rodola, "psutil documentation." [Online]. Available: https://psutil.readthedocs.io (accessed Oct. 2026).
[30]. Streamlit Inc., "Streamlit documentation." [Online]. Available: https://docs.streamlit.io (accessed Oct. 2026).

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