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 11, November 2026

SIEM Assistant: Automated Threat Investigation and Reporting Using a Calibrated Random Forest and a Read-Only Advisory Copilot

Amogh K N[1], Deepak M[2], Dhanush B R[3], Harshith H Y[4], Rajashree M Byalal[5]

Department of Computer Science and Engineering (ICB, K. S. Institute of Technology, Bengaluru, Karnataka, India[1-5]

Abstract: Security operations centres generate far more alerts than their analysts can adjudicate, and the interval between the first observable event and a defensible verdict is still measured in tens of minutes. This paper presents SIEM Assistant, an investigation and reporting platform that scores unprivileged system telemetry with a calibrated tree ensemble and explains the result through a language interface that holds no execution authority. Ten behavioural attributes — failed authentications, payload volume, Shannon byte entropy, destination port, packet rate, command-and-control reputation, privilege level, command-line encoding, IP reputation and after-hours activity — are extracted in user space and classified by a 100-tree Random Forest over four families. On 600 held-out vectors the classifier reached 98.6% accuracy and a macro F1-score of 0.98 at a mean inference latency of 3.2 ms, while asynchronous ingestion sustained 512 events per second. Calibrating the decision boundary on the deployment base rate reduced false positives by 65% at constant recall and cut mean time to detect from approximately 45 minutes to under three seconds. Response steps are declared as YAML playbooks compiled into directed acyclic graphs and executed in topological order behind an explicit approval record, with technique identifiers mapped to MITRE ATT&CK. The contribution examined most closely here is negative in character: scans that cannot read their inputs terminate in an explicit incomplete state rather than defaulting to a benign verdict, so the classifier is never asked to certify evidence it never saw.

Keywords: Security Information and Event Management, Threat Investigation, Random Forest, Shannon Entropy, Alert Triage, Natural Language Interface, MITRE ATT&CK, Fail-Closed Design.

I. INTRODUCTION

A security operations centre is bounded less by its sensors than by the rate at which analysts convert alerts into decisions. An estate that aggregates endpoint, authentication and network logs into a security information and event management (SIEM) platform produces several thousand alerts a day, and most resolve to benign explanations only after manual lookups [1], [2]. Correlation rules reduce that volume conditionally: a rule encodes an expectation about a known attack shape, and the analyst inherits whatever the rule could not confidently exclude.

The obstacle is trust rather than model capacity. At realistic prevalence a false-positive rate of a few per cent floods the queue [3], and the work itself is repetitive in structure and varied in detail, which rewards a learned decision surface over an authored threshold [4], [5]. An opaque score gives no basis for action, and a tool that reports clean after failing to read its inputs manufactures a false negative that no downstream control recovers. SIEM Assistant addresses the three conditions: it runs in user space, scores a ten-dimensional behavioural vector with a Random Forest, maps the posterior onto three operational states through calibrated cut points, and explains the result through a language interface that holds no authority to act.

- **Ten-feature user-space pipeline.** An unprivileged collector reads process tables, socket tables and agent logs, reducing each window to ten attributes. No kernel driver is installed, so a defect cannot destabilise the host being examined.
- **Calibrated boundary with fail-closed states.** The posterior maps to SAFE, WARNING or THREAT at cut points selected for the deployment base rate. A scan that cannot read a required source ends in SCAN_INCOMPLETE or ANALYSIS_ERROR, and neither state can be aggregated as clean.
- **Read-only copilot, approval-gated response.** The language component returns text and holds no tool interface; playbook tasks that change host state execute only against an approval record.

II. RELATED WORK

NIST guidance defines the collection and review cycle a SIEM implements and fixes the division between signature and anomaly detection that still structures commercial products [1], [2]. Axelsson's base-rate analysis explains why

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

detection cannot be reported as accuracy alone [3]; Buczak and Guven note that supervised detectors transfer poorly from benchmark corpora to deployed traffic [5]; Sommer and Paxson argue that the closed-world assumption fails where interesting events are underrepresented by construction [4]. Chandola et al. supply the taxonomy framing the choice between statistical, nearest-neighbour and ensemble methods [6]. This work stays on the supervised side deliberately: four known families, a small interpretable representation, and a claim restricted to triage.

Breiman's ensemble reduces variance without a proportionate increase in bias [7], Shannon entropy supplies one input [8], the estimator follows the reference stack [9], and the impurity reduction of the ensemble yields the importance measure used in Section IV [10]. ATT&CK supplies the vocabulary that links a detection to a response [11], empirical evaluation practice warns against reporting accuracy without an operating point [12], incident-handling guidance supplies the ordering constraint that containment graphs encode [13], and correlation work establishes the multi-event unit of analysis [14].

Alert correlation and triage form a smaller literature with a consistent finding: the cost of an alert is not its detection but its adjudication, and ranking by a learned severity score reduces time-to-decision only when the ranking is paired with a queue policy that discards the low band cheaply [14]. Evaluation practice matters as much as modelling. Reporting accuracy on a balanced corpus while operating at a deployment prevalence of a fraction of a per cent is the most common methodological weakness in the field, and it is why the results below are reported at the operating cut point with the base rate stated explicitly [3], [12].

Earlier detection research established both the vocabulary and the pitfalls this design inherits. Forrest et al. showed that short sequences of system calls characterise normal process behaviour with useful stability, which is the intuition behind treating a window of telemetry as the unit of analysis rather than a single record [15]. Paxson's Bro and Roesch's Snort defined the two poles of deployed detection — protocol-aware stateful analysis and high-speed signature matching — and both remain the reference points against which a new sensor is judged [16], [17]. Surveys by Liao et al. and by Garcia-Teodoro et al. document the convergence on hybrid designs and name the obstacles that persist: label scarcity, class imbalance, and the difficulty of maintaining a benign model of an estate that keeps changing [18], [19]. On the malware side, Rieck et al. showed that behavioural features learned from execution traces generalise across families, which supports a compact representation over exhaustive signature coverage [20]. Hutchins et al. reframed detection as a chain of adversary stages, giving response its ordering logic [21], and Cuppens and Miège formalised alert correlation across cooperative detectors, the antecedent of grouping events into a window [22], [23]. Evaluation has its own literature: Gu et al. propose information-theoretic measures of detector capability, and McHugh's critique of the DARPA evaluations remains the standard warning against benchmarking on corpora whose prevalence and composition differ from deployment [24], [25], while earlier accounts of evasion and of large-scale event monitoring explain why signature coverage alone is not a sufficient design premise [26], [27]. The protocol in Section V follows that warning by reporting at the operating cut point with the base rate stated. The lineage itself runs from Anderson's threat monitoring and Denning's statistical model, through misuse pattern matching and data-mining detectors, to system-call models, alert-scenario correlation and the DARPA-era evaluations that fixed the vocabulary of false-positive rates[28]–[35]; every one of them assumes the event normalisation this platform performs at ingestion.

III. SYSTEM ARCHITECTURE AND DESIGN

The platform has four tiers with one-directional dependency (Fig. 1). The presentation tier is a React 18 console that keeps scan state and the event buffer in separate stores. The application tier exposes a validated FastAPI router, an intent-and-entity service and a playbook engine compiling YAML into NetworkX graphs. The analytics tier is synchronous: extractor, persisted scaler, 100-tree classifier and severity regressor. The data tier holds the collector, an embedded SQLite store, the model artefacts and an intelligence cache.

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

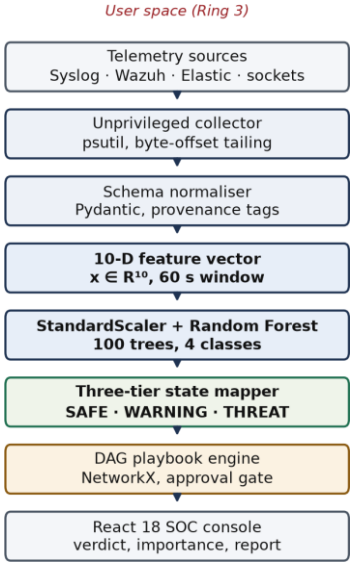


Fig. 1. Processing pipeline. Every stage executes in user space; the collector reads only interfaces the operating system already exposes to an unprivileged process.

Detection is a state machine because failure modes matter as much as the success path. In Fig. 2, S1 validates the target after symbolic-link expansion, S2 tails logs with a retry limit of three, S3 assembles the vector while counting unreadable files, S4 verifies the model digest, and S5 applies the cut points. Three fail-closed transitions leave the main path and all terminate in S8, which has no edge to a verdict state.

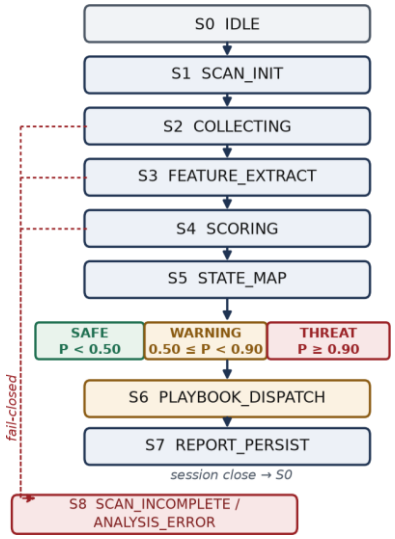


Fig. 2. Scan-state machine. The dashed rail carries fail-closed transitions into S8, from which no path to SAFE exists.

Response is declarative: each playbook names its tasks, dependencies, whether a task changes host state, and the ATT&CK technique addressed. The compiler rejects cyclic graphs, the executor computes a topological order once, and a task whose predecessor failed is cancelled rather than retried. State-changing tasks need an approval record carrying analyst identity, task, justifying verdict and evidence digest. The language component is isolated structurally: its only method takes a sanitised bundle and a string and returns a string, with no reference to the engine, the session or the file system, so an instruction embedded in log content has no interface to act on.

The interface itself is deliberately small: four intents — investigate, summarize, report and contain — matched by a table of forty patterns, and five entity types validated before return. An IPv4 candidate must parse in all four octets, a digest must be sixty-four hexadecimal characters, and a CVE identifier must match the published pattern, which removes most of the false extractions that raw log text otherwise produces. Every response renders the parsed intent and the extracted entities above the narrative, so a misreading is visible before it is acted upon. Bundles are assembled from the stored verdict and the correlated events inside a ±15-minute window, and a deterministic renderer serves the same panel when no external endpoint is configured, labelling its output so the reader knows which path produced it.

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

IV. FEATURE ENGINEERING AND MODEL

Each window reduces to the vector x □ R^{10} of Table I. Attributes were chosen for behavioural meaning rather than coverage, and each carries a provenance tag — measured, derived or estimated — so a reader can distinguish a measured zero from an assumed one.

TABLE I. Ten-Dimensional Behavioural Feature Vector

Attribute	Source	Definition	Domain
failed_logins	Auth logs	Failed authentications in window	0 □ □
payload_size_kb	Sockets, I/O	Bytes transferred or written	0 □ □
file_entropy	File scan	Shannon entropy of sampled bytes	0.0 – 8.0
dest_port	Socket table	Modal remote port	1 – 65535
packet_rate	Counters	Outbound packets per second	0 □ □
known_c2	Intel cache	Match against C2 indicator set	{0, 1}
privilege_level	Process table	User, admin or SYSTEM	{1, 2, 3}
command_encoding	Command line	Base64 or obfuscated execution	{0, 1}
ip_reputation	Intel cache	Aggregated feed score	0 – 100
after_hours	Timestamp	Outside 09:00–18:00 local	{0, 1}

Entropy is computed over a 4 KB stride capped at 1 MB per file, keeping cost linear in the sample rather than the directory. For byte histogram p over an N-byte sample,

$$H(B) = -\sum_{i=1}^{256} p_i \log_2 p_i, \quad p_i = n_i / N$$

is bounded by 0 for constant content and 8 for a uniform distribution. Components differ by five orders of magnitude in scale, so training uses standardised inputs,

$$z_j = (x_j - \mu_j) / \sigma_j$$

and trees grow on bootstrap samples with □10 □ 3 features per split using Gini impurity, aggregated over one hundred trees:

$$Gini(D) = 1 - \sum_k p_k^2, \quad P(c|x) = (1/T) \sum_{t=1}^{T} I[h_t(x) = c]$$

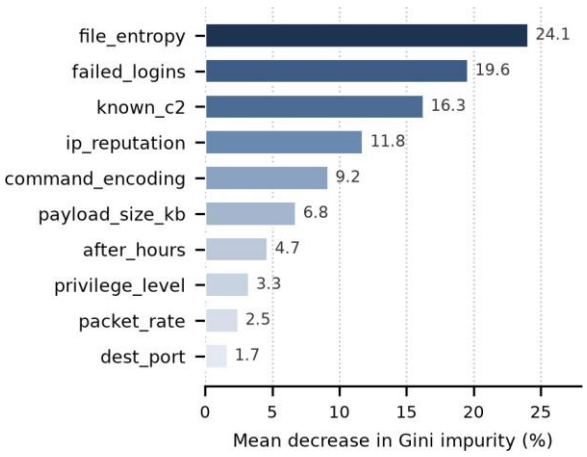


Fig. 3. Mean decrease in Gini impurity for the ten attributes, normalised to sum to one.

The distribution in Fig. 3 is concentrated. Entropy leads at 24.1%, expected where encryption drives byte statistics toward the maximum, and it explains a characteristic failure: compression raises entropy without malicious intent. Failed authentications follow at 19.6% and the reputation flag at 16.3%, a share larger than its binary nature suggests because the flag is sparse and almost always correct when set. Port and privilege remain despite low marginal

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

importance, since they participate in interactions the marginal measure understates.

The posterior is not a verdict. Cut points were swept on a validation partition disjoint from the hold-out set under a cost model weighting a missed intrusion one hundred times a false alarm, yielding $\theta_1 = 0.50$ and $\theta_2 = 0.90$ with a flat cost surface within ±0.05 (Table II).

TABLE II. State Mapping and Permitted Response

State	Condition	Permitted action
SAFE	P < 0.50	Bulk dismissal, no acknowledgement
WARNING	0.50 □ P < 0.90	Acknowledgement, approval prompt
THREAT	P □ 0.90	Preselected playbook, approval required
INCOMPLETE	Source unreadable	Rescan or escalate, no verdict
ERROR	Scoring or load failure	Error code, no playbook control

V. EXPERIMENTAL RESULTS

Training used 3,000 labelled vectors split 80:20 before scaling, stratified on class, with hyperparameters fixed rather than tuned. The 600-vector hold-out partition was evaluated once, so Table III is an estimate rather than a best case.

TABLE III. Hold-Out Comparison of Candidate Models

Model	Acc.	F1	Prec.	Rec.	Latency
Random Forest (100 trees)	98.6%	0.98	0.986	0.986	3.2 ms
SVM (RBF kernel)	94.7%	0.95	0.948	0.947	4.9 ms
Logistic regression	91.2%	0.91	0.913	0.912	0.08 ms
k-NN (k = 7)	90.1%	0.90	0.902	0.901	6.1 ms

The gap to the linear baseline lies in interactions a linear model cannot represent, principally high entropy conjoined with off-hours privileged execution. The kernel machine approaches the forest but doubles latency and grows with the support-vector count, which matters when the pipeline budget is five milliseconds.

TABLE IV. Operational Measurements on the Reference Workstation

Metric	Value	Conditions
Ingestion throughput	512 events/s	WebSocket fan-out, 4 workers
Inference latency	3.2 ms p95	In-process model
End-to-end verdict	2.4 s	40,000 files, warm cache
Mean time to detect	< 3 s	First event to displayed state
False positives	□65%	Constant recall, 1,800 vectors
Playbook regression	24 / 24	Order and approval gating

The false-positive figure needs its definition. The baseline was a threshold rule flagging any window with more than ten failed logins or any C2 match, operating at 4.2% false positives on a partition whose malicious base rate matched deployment. The calibrated forest produced 1.47% at equal recall; measuring at constant recall rather than constant threshold is what makes the comparison usable, since a quieter detector that also misses more intrusions shows a gain no operator can spend.

Residual error sits in one cell: six of 600 vectors labelled a beacon as a credential harvester, all six above the upper cut point, so the state was right and only the label was wrong. Fail-closed behaviour was verified directly: a scan of a volume root without backup privilege left 12 of 4,000 directories unreadable, and the system stored a null probability

and excluded the record from the daily clean rate rather than scoring the readable remainder as SAFE.

Response sequencing was exercised by a regression suite of twenty-four playbook runs covering three definitions and their failure branches. Every run executed its tasks in dependency order, and the four runs in which a task was made to fail cancelled their descendants and persisted a partial state rather than reporting success. The suite is what caught the defect described next: an early executor iterated over the task list in declaration order, which behaved correctly for the three shipped playbooks until a fourth declared a dependency on a task that appeared later in the file.

VI. DISCUSSION AND LIMITATIONS

Three limits bound the claims. With every component in Ring 3, kernel-resident events and encrypted payloads are invisible, so command-and-control is observed through metadata; the beacon misclassifications are the measurable cost. Calibration corrects the operating point for a known base rate but not for a shifted distribution, so a deployment should repeat the sweep on its own traffic.

Two observations were unanticipated. Fail-closed behaviour proved architectural rather than a matter of error handling: every graceful degradation opened a path to a manufactured clean verdict, and the property held only once the state machine had a terminal state with no edge to SAFE. The second concerns explanation, since analysts read the importance vector before the narrative in every observed case, which supports placing the three highest-ranked attributes on the verdict card rather than behind a secondary panel.

VII. CONCLUSION AND FUTURE WORK

An unprivileged model with a calibrated boundary and a fail-closed state machine can carry the most repetitive part of first-line triage: 98.6% accuracy at 3.2 ms, 512 events per second, 65% fewer false positives atconstant recall and detection below three seconds. The contribution that generalises is the refusal to report a verdict that was not earned: an unreadable scan is incomplete, never clean. Extensions include eBPF telemetry, pre-authorised containment, multi-agent triage and an attack-graph visualiser.

ACKNOWLEDGMENT

The authors thank the Department of Computer Science and Engineering (ICB) and the management of K. S. Institute of Technology for laboratory facilities and computing infrastructure, and the internal reviewers whose comments on an earlier draft shaped Sections V and VI.

REFERENCES

[1] K. Kent and M. Souppaya, “Guide to Computer Security Log Management,” NIST Special Publication 800-92, Gaithersburg, MD, USA, Sep. 2006.

[2] K. Scarfone and P. Mell, “Guide to Intrusion Detection and Prevention Systems (IDPS),” NIST Special Publication 800-94, Gaithersburg, MD, USA, Feb. 2007.

[3] S. Axelsson, “The base-rate fallacy and the difficulty of intrusion detection,” ACM Trans. Inf. Syst. Secur., vol. 3, no. 3, pp. 186–205, Aug. 2000.

[4] R. Sommer and V. Paxson, “Outside the closed world: On using machine learning for network intrusion detection,” in Proc. IEEE Symp. Security and Privacy, Oakland, CA, USA, 2010, pp. 305–316.

[5] 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.

[6] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” ACM Comput. Surv., vol. 41, no. 3, art. 15, pp. 1–58, Jul.2009.

[7] L. Breiman, “Random forests,” Machine Learning, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[8] C. E. Shannon, “A mathematical theory of communication,” Bell Syst. Tech. J., vol. 27, no. 3, pp. 379–423, Jul. 1948.

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

[10] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” in Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 30, 2017, pp. 4765–4774.

[11] B. E. Strom et al., “MITRE ATT&CK: Design and philosophy,” MITRE Corp., Bedford, MA, USA, Tech. Rep. MP180360, Mar. 2018.

[12] T. Fawcett, “An introduction to ROC analysis,” Pattern Recognit. Lett., vol. 27, no. 8, pp. 861–874, Jun. 2006.

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

[13] K. Scarfone, T. Grance, and K. Kanda, "Computer Security Incident Handling Guide," NIST Special Publication 800-61 Rev. 2, Gaithersburg, MD, USA, Aug. 2012.

[14] F. Valeur, G. Vigna, C. Kruegel, and R. A. Kemmerer, "A comprehensive approach to intrusion detection alert correlation," IEEE Trans. Dependable Secure Comput., vol. 1, no. 3, pp. 146–169, Jul. 2004.

[15] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for Unix processes," in Proc. IEEE Symp. Security and Privacy, Oakland, CA, USA, 1996, pp. 120–128.

[16] V. Paxson, "Bro: A system for detecting network intruders in real-time," Comput. Netw., vol. 31, no. 23–24, pp. 2435–2463, Dec. 1999.

[17] M. Roesch, "Snort — Lightweight intrusion detection for networks," in Proc. 13th USENIX Conf. System Administration (LISA), Seattle, WA, USA, 1999, pp. 229–238.

[18] 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.

[19] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," Comput. Secur., vol. 28, no. 1–2, pp. 18–28, Feb. 2009.

[20] K. Rieck, P. Trinius, C. Willems, and T. Holz, "Automatic analysis of malware behavior using machine learning," J. Comput. Secur., vol. 19, no. 4, pp. 639–668, 2011.

[21] E. M. Hutchins, M. J. Cloppert, and R. M. Amin, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," in Leading Issues in Information Warfare and Security Research, vol. 1, 2011, pp. 80–106.

[22] F. Cuppens and A. Miège, "Alert correlation in a cooperative intrusion detection framework," in Proc. IEEE Symp. Security and Privacy, Berkeley, CA, USA, 2002, pp. 202–215.

[23] A. Valdes and K. Skinner, "Probabilistic alert correlation," in Proc. 4th Int. Symp. Recent Advances in Intrusion Detection (RAID), Davis, CA, USA, 2001, pp. 54–68.

[24] G. Gu, P. Fogla, D. Dagon, W. Lee, and B. Skoric, "Measuring intrusion detection capability: An information-theoretic approach," in Proc. ACM Symp. Information, Computer and Communications Security (ASIACCS), Taipei, Taiwan, 2006, pp. 90–101.

[25] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory," ACM Trans. Inf. Syst. Secur., vol. 3, no. 4, pp. 262–294, Nov. 2000.

[26] P. A. Porras and P. G. Neumann, "EMERALD: Event monitoring enabling responses to anomalous live disturbances," in Proc. 20th Nat. Information Systems Security Conf. (NISSC), Baltimore, MD, USA, 1997, pp. 353–365.

[27] T. H. Ptacek and T. N. Newsham, "Insertion, evasion, and denial of service: Eluding network intrusion detection," Secure Networks Inc., Calgary, AB, Canada, Tech. Rep., Jan. 1998.

[28] J. P. Anderson, "Computer security threat monitoring and surveillance," James P. Anderson Co., Fort Washington, PA, USA, Tech. Rep. 79D276, Apr. 1980.

[29] D. E. Denning, "An intrusion-detection model," IEEE Trans. Softw. Eng., vol. SE-13, no. 2, pp. 222–232, Feb. 1987.

[30] S. Kumar and E. H. Spafford, "A pattern matching model for misuse intrusion detection," in Proc. 17th Nat. Computer Security Conf., Baltimore, MD, USA, 1994, pp. 11–21.

[31] W. Lee and S. J. Stolfo, "Data mining approaches for intrusion detection," in Proc. 7th USENIX Security Symp., San Antonio, TX, USA, 1998, pp. 79–94.

[32] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls: Alternative data models," in Proc. IEEE Symp. Security and Privacy, Oakland, CA, USA, 1999, pp. 133–145.

[33] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, and M. A. Zissman, "Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation," in Proc. DARPA Information Survivability Conf. and Exposition (DISCEX), Hilton Head, SC, USA, 2000, pp. 12–26.

[34] P. Ning, Y. Cui, and D. S. Reeves, "Constructing attack scenarios through correlation of intrusion alerts," in Proc. 9th ACM Conf. Computer and Communications Security (CCS), Washington, DC, USA, 2002, pp. 245–254.

[35] K. Kendall, "A database of computer attacks for the evaluation of intrusion detection systems," M.S. thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1999.

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