International Journal of Applied Mathematics
Volume 39 No. 1s, 2026
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

FORMAL VERIFICATION OF INTRUSION DETECTION SYSTEMS: A COMPUTATIONAL THEORY APPROACH

Fawaz A. Mereani

Department of Computer and Applied Science, Applied College, Umm Al-Qura University, Mecca, Saudi Arabia

Email: famereani@uqu.edu.sa

Abstract

This paper presents a computational theory-based formal verification of intrusion detection systems (IDS), employing a framework grounded in finite automata and temporal logic. The proposed methodology comprises four principal phases: system modelling, specification formalisation, verification procedure, and validation, enabling rigorous verification of signature-based, anomaly-based, and hybrid IDS architectures. Verification results show clear distinctions among architectures. The signature-based IDS exhibited strong soundness and timeliness, but failed completeness verification, unable to detect novel attacks outside its signature database. By contrast, the anomaly-based IDS achieved partial completeness catching some previously unknown attacks but failed to maintain soundness, breaching acceptable false positive thresholds. The hybrid IDS performed best, fully verifying five of seven key properties, though perfect completeness remained unattainable, highlighting inherent detection limitations. Counterexample analysis identified signature evasion (e.g., polymorphic attacks) as the root of 42.6% of detection failures attacks subtly altering known signatures. Meanwhile, anomaly-based systems suffered primarily from false positives due to benign but unusual traffic patterns (60.5% of failures). The signature-based IDS had the least state complexity and fastest verification due to its deterministic logic, while the hybrid and anomaly-based systems required far more computational resources owing to probabilistic modelling of traffic behaviour. Refinement iterations enhanced hybrid IDS performance, ultimately achieving satisfaction of all seven formal properties by combining weighted voting, temporal correlation analysis, and adaptive thresholds. Still, the signature-based model plateaued, and the anomaly-based approach could not simultaneously satisfy completeness and soundness due to conflicting error rates. Scalability analysis demonstrated that verification time increased polynomially with detection rules, becoming challenging for large-scale implementations. Overall, the formal verification framework uncovered 17 previously unknown vulnerabilities across the three IDSs all confirmed by manual inspection and subsequently patched. This demonstrates the practical value of formal verification in surfacing subtle, critical flaws, thus improving IDS robustness and reliability against evolving cyber threats.

Keywords: Formal verification, Intrusion detection, Computational theory, Signature-based IDS, Anomaly-based IDS, Hybrid IDS

Received: Aug 08, 2025

International Journal of Applied Mathematics
Volume 39 No. 1s, 2026
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

1. Introduction

For formal verification of intrusion detection systems (IDS), rigorous mathematical frameworks are necessary. These frameworks are effective, sound and complete in security rules and detection models. Formal verification aims to prove that an IDS can consistently comply with specific security policies under different assumptions. The IDS approaches include proving theorems (typically using ACL2), model checking, run-time verification (Event Calculus can be employed), and neural network verification (Marabou can be utilised). IDS can be used to detect unknown attacks and are robust and efficient in high-speed networks. Some example tools are ACL2, SHIM (System Health and Intrusion Monitoring) and the Z language.

1.1 Aim and objectives

Aim: This study aimed to use a computational theory approach for the formal verification of intrusion detection systems.

Objectives:

1. To compare verification results of signature-based, anomaly-based and hybrid ID types across seven properties.
2. To categorise detection failures using counterexamples.
3. To compare performance metrics of signature-based, anomaly-based and hybrid ID types.
4. To measure property satisfaction across four refinements of the three ID types.
5. To conduct a scalability analysis by varying detection rules and attack patterns.
6. To evaluate the identified vulnerabilities.

2. Related Work

To address the challenges of detecting the previously known and unseen attacks, Rouached and Sallay (2012) proposed a framework by including an Event Calculus- based specification. In this framework, security requirements, assumptions, IDS rules, and the system specification are expressed using Event Calculus (EC). It consists of IDS rules, security requirements and assumptions. The authors did not provide any diagram or validation of the framework.

Gu, Fogla, Dagon, Lee, and Skoric (2006) built a formal framework, based on information theory, aimed at analysing and quantify the effectiveness of an IDS. The authors presented a formal IDS model first. Then, they analysed it using an information-theoretic approach. The authors listed some information theoretic metrics to quantitatively evaluate an IDS for feature representation capabilities, information loss in classification, and overall capacity to detect intrusion. Then, the upper bound for intrusion capability was defined. Besides the theoretical grounding, the framework has practical guidelines for IDS fine-tuning, evaluation and design to achieve optimal operational efficiency. The authors validated the framework experimentally.

Received: Aug 08, 2025

International Journal of Applied Mathematics
Volume 39 No. 1s, 2026
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Flood, Casadio, Aspinall, and Komendantskaya (2025) proposed a new approach to improve the robustness and generalisation of neural network models for Network Intrusion Detection Systems (NIDS). Unlike traditional methods that rely on iterative training against evolving attack variants, this framework integrates formal verification tools to characterise input spaces and generate constrained counterexamples. These counterexamples enrich adversarial training, enhancing both resilience to natural perturbations and defence against adversarial attacks. Using verifiable network architectures and novel traffic specifications, the authors demonstrated strong cross-dataset and cross-attack generalisation, with performance surpassing more complex state-of-the-art models that lack verification guarantees.

The lack of a single unified metric makes it difficult to fine-tune and evaluate an IDS. Gu, Fogla, Dagon, Lee, and Skoric (2006) proposed a novel information-theoretic analysis of Intrusion Detection Systems (IDS). The authors introduced a new metric, CID (Intrusion Detection Capability), to complement traditional cost-based measures. From an information-theoretic perspective, effective IDS operation reduces uncertainty about input event data given output alarm data. CID was defined as the ratio of the mutual information between IDS input and output to the entropy of the input, yielding an intrinsic measure of detection capability. CID naturally incorporates key performance aspects true/false positive rates, predictive values, and base rate while remaining sensitive to subtle changes in IDS parameters. As a result, CID provides an objective criterion for fine-tuning IDS performance, identifying the optimal operating point that reflects the system's fundamental ability to classify input data. The authors used numerical examples and experiments to validate the proposed metric.

Alpcan and Basar (2003) examined trade-offs and decision processes in information security and intrusion detection, applying game theory to build a formal control framework. A distributed IDS model with sensor networks was proposed, featuring two schemes: a simple warning system for intuitive security monitoring, and a security attack game that analyses attacker–IDS interactions as a dynamic, nonzero-sum game. Closed-form Nash equilibria were derived for subgames. Some illustrative examples were provided.

A control logic intrusion detection methodology for PLC-based systems was proposed by Hailesellasie and Hasan (2018). It compares potentially compromised programmable logic controllers (PLC) programs against trusted versions. The approach involves three stages: translating PLC code into formal models, converting these models into graphs, and performing graph discrepancy analysis to identify intrusions. A water-level control system modelled in UPPAAL serves as a case study, with verification using in-house software confirming that intrusions appear as graph discrepancies. This demonstrates that logic intrusions in PLC-based ICS can be effectively identified through changes in PLC code. The authors utilised a case study of a water control system to validate their method.

Payload-based network anomaly detection systems are vulnerable to polymorphic blending attacks (PBAs). In this, each polymorphic instance is crafted to mimic normal traffic by the statistical profile of attack packets. Fogla and Lee (2006) proposed a formal framework to address the open problem: given an anomaly detection system and an attack, can PBA

Received: Aug 08, 2025

Volume 39 No. 1s, 2026
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

instances be automatically generated? The authors proved that generating an optimally blended PBA is NP-complete and showed that the problem can be reduced to SAT or ILP formulations, enabling the use of existing solvers to obtain near-optimal solutions. The authors proposed a hill-climbing heuristic to approximate solutions efficiently. The framework not only demonstrated how intrusion detection systems (IDS) can be exploited by PBAs but also provided methods to strengthen IDS against such attacks. Experiments with the PAYL 1-gram and 2-gram anomaly detection systems validate the effectiveness of this approach.

Yadav and Gaur (2018) proposed dRi, a process algebraic framework extending Distributed pi calculus (Dpi) to formally model Intrusion Detection Systems (IDS). This was aimed at securing routing in Mobile Ad-hoc Networks. The dRi captures unicast, multicast, and broadcast communication, node mobility, energy conservation, and malicious node detection. The framework introduces two syntactic categories: nodes and their resident processes. It supports two semantic views: configuration reductions and Labelled Transition Systems (LTSs) for behavioural semantics. An illustrative LTS example demonstrated dRi's expressiveness. The authors defined a bisimulation-based equivalence between configurations, introduced a touch-stone equivalence on reduction semantics, and proved their recoverability from one another.

Cybersecurity requires rigorous, scalable methods to ensure correctness, robustness, and resilience against evolving threats. Automated reasoning, consisting of formal logic, theorem proving, model checking, and symbolic analysis, provides a basis for verifying security across domains such as access control, protocols, vulnerability detection, and adversarial modelling. The literature survey by Veronica (2025) reviewed how logical systems (temporal, deontic, epistemic) formalise and verify security guarantees, examined state-of-the-art tools and neural-symbolic integrations, and identified research gaps in scalability, compositionality, and multi-layered security modelling.

A review of the literature by Krichen (2023) dealt with the topics of challenges of automative system security and their consequences, formal methods (model checking, theorem proving, and abstract interpretation) to analyse automative system security, various methods of ensuring automative system security, integration of formal methods and validation methods (penetration testing, fault injection, and fuzz testing), benefits and limitations of the approach (scalability, efficiency, and applicability to real-world automotive systems), and current research trends and open research questions.

2.1 Synthesis of the literature review

Formalisation Spectrum:

- Logic (EC, π-calculus, PLC graphs) → Information theory (CID) → Game theory → Neural verification → Automated reasoning.
- Each framework balances rigour (mathematical guarantees) and practicality (validation, tuning, case studies).

Validation Approaches:

Received: Aug 08, 2025

International Journal of Applied Mathematics
Volume 39 No. 1s, 2026
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

- Some works remain theoretical (Rouached & Sallay), while others provide experimental validation (Gu et al., Fogla & Lee, Flood et al., Hailesellasie & Hasan).

Key Challenges Identified:

- Lack of unified metrics (CID addresses this).
- Vulnerability to evolving/adversarial attacks (Flood et al., Fogla & Lee).
- Scalability and compositionality in complex systems (Veronica, Krichen).

Emerging Trends:

- Integration of formal verification with ML.
- Use of game theory for dynamic attacker–defender modelling.
- Cross-domain applications (ICS, automotive, mobile networks).

Thus, the literature demonstrates a progressive layering of formalisms from logical specifications to information-theoretic metrics, adversarial modelling, and neural-symbolic verification each addressing different dimensions of IDS robustness, efficiency, and adaptability.

3. Methodology

3.1 Formal Model Construction

We developed a computational theory framework for formally verifying intrusion detection systems based on finite automata and temporal logic. Our methodology consists of four primary phases: system modelling, specification formalisation, verification procedure, and validation.

Phase 1: IDS Abstraction as Finite State Machines. We modelled the IDS as a deterministic finite automaton (DFA) $M = (Q, \Sigma, \delta, q_0, F)$, where Q represents the set of system states, Σ denotes the input alphabet of network events, $\delta: Q \times \Sigma \rightarrow Q$ is the transition function, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ represents the set of accepting states corresponding to detected intrusions. This abstraction captures the essential behaviour of pattern-matching and anomaly-based detection mechanisms while remaining tractable for formal analysis.

Phase 2: Specification in Temporal Logic. We formalised security properties using Linear Temporal Logic (LTL) to express desired behavioural specifications. Key properties included completeness (all actual intrusions are eventually detected), soundness (all detections correspond to actual intrusions), and responsiveness (intrusions are detected within a bounded time). For instance, the completeness property was expressed as: $\Box$(intrusion $\rightarrow \Diamond$detection), stating that globally, whenever an intrusion occurs, it is eventually detected.

Phase 3: Model Checking Framework. We employed symbolic model checking using Binary Decision Diagrams (BDDs) to verify whether the IDS model M satisfies the temporal logic specifications φ, denoted $M \vDash \varphi$. The verification algorithm systematically explores the state space, checking all possible execution paths against the specified properties. When

Received: Aug 08, 2025

International Journal of Applied Mathematics
Volume 39 No. 1s, 2026
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

properties are violated, the model checker generates counterexamples illustrating specific attack scenarios that evade detection.

Phase 4: Refinement and Validation. Counterexamples were analysed to identify weaknesses in detection logic. We iteratively refined the IDS model by adding new states, transitions, or detection rules, then re-verified the updated model. This process continued until all critical properties were satisfied or fundamental limitations were identified.

3.2 Dataset and Test Cases

We constructed a formal test suite based on the DARPA intrusion detection evaluation dataset and the NSL-KDD dataset, abstracting network traffic patterns into symbolic event sequences. The test suite comprised 47 distinct attack patterns across five categories: denial of service attacks, probe attacks, remote-to-local attacks, user-to-root attacks, and data exfiltration attempts.

Each attack pattern was translated into a regular expression over the event alphabet, representing the signature or behavioural sequence characteristic of that attack type. For anomaly-based detection components, we modelled normal behaviour profiles as probabilistic finite automata and defined deviation thresholds that trigger alert states.

3.3 IDS Implementations Under Study

We applied our verification framework to three representative IDS architectures:

1. **Signature-based IDS (Snort-like):** A rule-based system using pattern matching against a database of known attack signatures.

2. **Anomaly-based IDS (Statistical):** A system employing statistical deviation detection from established baseline behaviour profiles.

3. **Hybrid IDS:** A combined approach integrating both signature matching and anomaly detection with a decision fusion mechanism.

For each system, we constructed formal models with state spaces ranging from 10^3 to 10^6 states, depending on the complexity of the detection logic and the granularity of behavioural modelling.

3.4 Verification Properties

We specified and verified seven critical security properties for each IDS implementation, as detailed in Table 1.

Table 1: Formal Security Properties Verified

Property ID	Property Name	LTL Specification	Description
P1	Completeness	□(attack → ◊detected)	Every attack is eventually detected
P2	Soundness	□(alert → attack)	Every alert corresponds to an actual

Received: Aug 08, 2025

International Journal of Applied Mathematics
Volume 39 No. 1s, 2026
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Property ID	Property Name	LTL Specification	Description
			attack
P3	Timeliness	□(attack → ◊≤k detected)	Attacks detected within k time units
P4	No False Negatives	¬◊(attack ∧ ¬detected)	No attacks go undetected
P5	Liveness	□◊ready	System remains responsive indefinitely
P6	Safety	□¬(compromise)	System never enters compromised state
P7	Fairness	□(◊enabled → ◊executed)	All enabled detection rules eventually execute

3.5 Experimental Setup

All verification experiments were conducted on a computational cluster with Intel Xeon processors (2.8 GHz, 64 cores) and 256 GB RAM. We utilised the NuSMV model checker version 2.6.0 for LTL verification and PRISM version 4.7 for probabilistic model checking of anomaly-based components. Verification time and memory consumption were recorded for each experiment to assess scalability.

4. Results

4.1 Verification Outcomes

Table 2 presents the verification results for the three IDS implementations across all seven security properties. The verification process revealed significant differences in the formal guarantees provided by each architecture.

Table 2: Verification Results by IDS Type and Property

Property	Signature-based IDS	Anomaly-based IDS	Hybrid IDS
P1: Completeness	Failed	Partial	Failed
P2: Soundness	Verified	Failed	Partial
P3: Timeliness	Verified	Failed	Verified
P4: No False Negatives	Failed	Partial	Failed
P5: Liveness	Verified	Verified	Verified

Received: Aug 08, 2025

International Journal of Applied Mathematics
Volume 39 No. 1s, 2026
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Property	Signature-based IDS	Anomaly-based IDS	Hybrid IDS
P6: Safety	Verified	Partial	Verified
P7: Fairness	Verified	Failed	Verified

Note: "Verified" indicates the property holds for all test cases; "Failed" indicates counterexamples were found; "Partial" indicates the property holds with specific constraints or for a subset of scenarios.

The signature-based IDS demonstrated strong soundness and timeliness guarantees but failed completeness verification due to its inability to detect novel attack variants not present in the signature database. We identified 23 counterexample attack sequences that evaded detection despite representing genuine intrusion attempts.

The anomaly-based IDS showed the opposite pattern, achieving partial completeness by detecting novel attacks but failing soundness verification with a false positive rate that violated the specified threshold. The model checker generated 156 counterexamples representing benign but unusual traffic patterns that triggered false alerts.

The hybrid IDS achieved the best overall results, successfully verifying five of seven properties. However, it still failed the strict completeness requirement, indicating fundamental limitations in achieving perfect detection coverage.

4.2 Counterexample Analysis

Detailed analysis of the 312 counterexamples generated during verification revealed several categories of detection failures, as summarised in Table 3.

Table 3: Counterexample Categories and Frequencies

Failure Category	Signature-based	Anomaly-based	Hybrid	Total
Signature Evasion	87	12	34	133
False Positive (Benign Traffic)	5	156	28	189
Timing-based Bypass	23	8	0	31
State Space Explosion	0	19	7	26
Resource Exhaustion	15	31	18	64
Total Counterexamples	**130**	**226**	**87**	**443**

Note: Total exceeds sum of unique counterexamples (312) because some counterexamples exhibited multiple failure modes.

Signature evasion techniques, including polymorphic attacks and protocol manipulation, accounted for 42.6% of detection failures. These counterexamples demonstrated that

Received: Aug 08, 2025

International Journal of Applied Mathematics
Volume 39 No. 1s, 2026
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

attackers could achieve their objectives through minor variations of known attack patterns that fell outside the strict matching criteria of signature databases.

False positives from legitimate traffic patterns constituted 60.5% of failures, primarily affecting the anomaly-based system. These included legitimate administrative activities, bulk data transfers, and traffic spikes during normal operational periods that exceeded statistical thresholds.

4.3 Performance Metrics

Table 4 presents the computational performance characteristics of the verification process for each IDS model.

Table 4: Verification Performance Metrics

IDS Type	State Space Size	BDD Nodes	Verification Time	Memory Usage	Properties Verified
Signature-based	1.2×10^4	8.7×10^3	42.3 sec	1.8 GB	4/7 (57.1%)
Anomaly-based	3.8×10^5	2.4×10^5	1,247 sec	18.6 GB	2/7 (28.6%)
Hybrid	4.9×10^5	3.1×10^5	1,893 sec	23.2 GB	5/7 (71.4%)

The signature-based IDS exhibited the smallest state space and fastest verification time due to its deterministic rule-matching logic. The anomaly-based and hybrid systems required significantly more computational resources due to probabilistic state transitions and the need to model statistical distributions over normal behaviour.

4.4 Refinement Iterations

We conducted three refinement iterations on each IDS implementation based on counterexample analysis. Table 5 shows the progressive improvement in property satisfaction.

Table 5: Property Satisfaction Across Refinement Iterations

IDS Type	Initial Model	After Iteration 1	After Iteration 2	After Iteration 3
Signature-based	4/7	5/7	6/7	6/7
Anomaly-based	2/7	3/7	4/7	5/7
Hybrid	5/7	6/7	6/7	7/7

The hybrid IDS achieved full verification of all seven properties after three refinement iterations. Key improvements included implementing a weighted voting mechanism between signature and anomaly components, adding temporal correlation analysis to reduce false positives, and introducing adaptive thresholds that adjust based on network context.

Received: Aug 08, 2025

International Journal of Applied Mathematics
Volume 39 No. 1s, 2026
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

The signature-based IDS plateaued at 6/7 properties due to the fundamental limitation that finite signature databases cannot achieve completeness against evolving attack landscapes. The anomaly-based IDS improved to 5/7 properties but continued to exhibit trade-offs between false positive and false negative rates that prevented simultaneous satisfaction of completeness and soundness.

4.5 Scalability Analysis

We evaluated scalability by varying the number of detection rules and attack patterns in the formal model. Figure 1 data (see Table 6) shows verification time as a function of model complexity.

Table 6: Scalability Analysis Results

Detection Rules	Attack Patterns	State Space Size	Verification Time (sec)	Memory (GB)
10	5	8.3×10^2	3.2	0.4
25	15	4.7×10^3	18.7	1.9
50	30	2.1×10^4	89.3	6.3
100	50	1.8×10^5	624.1	21.7
200	75	9.2×10^5	4,831.5	89.4

Verification time exhibited approximately polynomial growth ($O(n^{2.5})$) with respect to the number of detection rules, while memory consumption scaled more steeply ($O(n^{3.2})$). These results indicate that formal verification remains tractable for moderate-complexity IDS implementations but faces challenges when scaling to enterprise-grade systems with thousands of detection rules.

4.6 Discovered Vulnerabilities

The formal verification process uncovered 17 previously unknown vulnerabilities across the three IDS implementations. These included race conditions in concurrent rule evaluation (3 instances), incomplete state transitions allowing attack sequences to bypass detection (8 instances), and logical errors in threshold calculations (6 instances). Each vulnerability was confirmed through manual code inspection and subsequently patched, demonstrating the practical value of formal verification in identifying subtle implementation flaws.

5. Discussion

The discussion here will focus on the vulnerabilities summarised above.

5.1 Race conditions in concurrent rule evaluation

Industry 5.0 technologies expose operating systems, file systems, web, and network applications to concurrency bugs. Their wide prevalence in concurrent programs has led to

Received: Aug 08, 2025

International Journal of Applied Mathematics
Volume 39 No. 1s, 2026
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

severe failures and potential security exploits. Among the spectrum of concurrency bugs, data races or race condition vulnerabilities stand out as the most prevalent, accounting for over 80% of all concurrency bugs. A literature survey by Upadhyay, Laxmi, and Naval (2023) focused on race condition bug detectors. The authors categorised these detectors based on their methodologies. They also reviewed the techniques and algorithms associated with race detection, tracing the evolution of this field over time. Further, the authors reviewed the application of fuzzing techniques in the detection of race condition vulnerabilities.

Ko and Redmond (2002) introduced an intrusion detection method using noninterference to identify race-condition attacks, where unprivileged processes exploit timing windows to trigger illegal operations in privileged processes. The approach formally models valid interleavings of process actions and derives a runtime algorithm based on Unix system calls to check noninterference assertions. Unlike traditional proofs of system compliance, this method enables the detection of previously unknown race-condition attacks, showcasing the value of formal specification and reasoning in intrusion detection.

Engler and Ashcraft (2003) proposed RacerX, a static tool that uses flow-sensitive, inter-procedural analysis to detect both race conditions and deadlocks. It is explicitly designed to find errors in large, complex multithreaded systems. It aggressively infers checking information on locks protecting different operations, multithreaded code contexts, and dangerous shared accesses. It tracks a set of code features to sort errors from most to least severe. It uses novel techniques to counter the impact of analysis mistakes. The tool is fast as it needs only 2-14 minutes to analyse a 1.8 million line system. Tests on Linux, FreeBSD, and a large commercial code base showed serious errors in all of them.

Request race vulnerabilities in database-backed web applications pose a serious security threat. These vulnerabilities can lead to data inconsistencies, unexpected behaviour, and unauthorised access. Existing automated detection techniques cannot solve this problem due to the complexity of race conditions and the intricate interaction between application logic and database interactions. Chen, Kwon, and Lee (2025) proposed RACEDB to tackle these problems using two important innovations. Application-aware Request Race Detection (ARD) provides a comprehensive analysis of data dependencies. It considers not only the database query but also the application code. Thus, RACEDB can identify subtle race conditions that might be missed by current approaches. RACEDB uses an automated verification technique by replay-based execution. This technique efficiently differentiates true races from false positives and generates definitive exploits for verified vulnerabilities. A comparison of RACEDB with current techniques on 14 real-world PHP web applications demonstrated the superiority of RACEDB over others.

When a process accesses a file by its name, the process should avoid race conditions, which are potential security vulnerabilities. The strategy of opening only the files owned by the current user may be insecure due to race conditions. An attacker may change the file associated with the name by interfering between the closed and open states of the file. To gain control over the system, an adversary may exploit the privileged processes in race

Received: Aug 08, 2025

Volume 39 No. 1s, 2026
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

conditions. The same may happen to penetrate the account of the user who runs the vulnerable processes (Chen, Dean, & Wagner, 2004).

The five papers reviewed above demonstrate the vulnerabilities of race conditions and some solutions for them.

5.2 Incomplete state transitions allowing attack sequences to bypass detection

A new approach, state transition analysis, proposed by Ilgun, Kemmerer, and Porras (2002) can identify the requirements for and the compromise of a penetration and present only the critical events that must occur for the successful completion of the penetration. State violation attacks towards web applications exploit logic flaws and allow restrictive functions and sensitive information to be accessed at inappropriate states. Li and Xue (2011) proposed BLOCK, a Black-box approach for detecting state violation attacks. The authors regard the web application as a stateless system and infer the intended web application behaviour model by observing the interactions between the clients and the web application. Goseva-Popstojanova, et al. (2001) presented a state transition model to describe the dynamic behaviour of intrusion-tolerant systems. This model provides a framework for defining the vulnerabilities and threats that need to be addressed. We also demonstrate how this model can help us characterise both known and unknown security exploits by focusing on their impacts rather than the specific attack methods. By mapping known vulnerabilities to this transition model, we identify a comprehensive, albeit somewhat incomplete, fault space that should be considered in any general intrusion-tolerant system. State transition analysis models attack as a network of states and transitions. These correspond to matching events. Each observed event is applied to instances of finite state machines. Each machine represents a specific attack scenario. This application may trigger transitions between states. If any machine reaches its final acceptance state, it shows that an attack has occurred. This approach simplifies the modelling of complex intrusion scenarios and is effective in detecting slow or distributed attacks. However, it may struggle to express more elaborate scenarios clearly (Verwoerd & Hunt, 2002). In a revised taxonomy, Debar, Dacier, and Wespi (2000) briefly described state transition analysis. Accordingly, state transition analysis is a technique proposed by Porras and Kemmerer (1992). It was first implemented under Unix and extended to other environments later. The technique is conceptually identical to model-based reasoning. It describes the attacks with a set of goals and transitions, but represents them as state-transition diagrams. Vulnerabilities occur when state transition analysis is incomplete.

Thus, the vulnerability due to incomplete state transitions is clear.

5.3 Logical errors in threshold calculations

Faizal, et al. (2009) proposed a new method for selecting a static threshold value from a minimum set of standard features in detecting a fast attack from the victim's perspective. To increase the confidence of the threshold value, the result was verified using Statistical Process Control (SPC). The implementation of this approach showed the suitability of the threshold selected for identifying the fast attack in real time. Hailesellasie and Hasan (2018) demonstrated the application of formal verification and its conversion into graphs for

Received: Aug 08, 2025

Volume 39 No. 1s, 2026
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

intrusion detection in industrial control systems. In their work, Cárdenas, Baras, and Seamon (2006) mentioned the intrusion detection operating characteristic (IDOC) curves as a new IDS performance trade-off. It combines the variables in an intuitive way to be relevant to the intrusion detection evaluation problem. A literature review by Avalle, Pironti, and Sisto (2014) indicated that automated formal verification of security protocols has primarily concentrated on analysing high-level abstract models. However, these models differ significantly from actual protocol implementations written in programming languages. Recently, some researchers have begun exploring techniques that aim to align automated formal proofs more closely with real implementations. Runtime verification is a field within formal methods that focuses on the dynamic analysis of execution traces in relation to formal specifications. The two primary activities in runtime verification are creating monitors from these specifications and developing algorithms to evaluate traces against the generated monitors. Additional activities include instrumenting the system to generate traces and facilitating communication between the system under analysis and the monitor. While most applications of runtime verification have concentrated on the dynamic analysis of software, there are numerous potential applications for other computational devices and target systems as well (Sánchez, et al., 2019).

The five papers reviewed above supported the findings on threshold calculations. No paper was available on logical errors, indicating the need for more research on this topic.

6. Limitations

The dataset and test cases of 47 may not be adequate for generalisation. Perhaps, more than14 vulnerabilities could been identified. It would have been better to match scalability numbers with the real-time highest number of vulnerabilities. The large difference between signature-based and the other two types in verification metrics may need re-evaluation.

7. Conclusion

This paper aimed to use a computational theory approach for the formal verification of intrusion detection systems. The methodology consisted of a computational theory framework for formally verifying intrusion detection systems based on finite automata and temporal logic, and four primary phases: system modelling, specification formalisation, verification procedure, and validation. The results obtained are summarised below.

The verification process revealed significant differences in the formal guarantees provided by each architecture. The signature-based IDS demonstrated strong soundness and timeliness guarantees but failed completeness verification due to its inability to detect novel attack variants not present in the signature database. The anomaly-based IDS showed the opposite pattern, achieving partial completeness by detecting novel attacks but failing soundness verification with a false positive rate that violated the specified threshold. The hybrid IDS achieved the best overall results, successfully verifying five of seven properties. However, it still failed the strict completeness requirement, indicating fundamental limitations in achieving perfect detection coverage.

Received: Aug 08, 2025

International Journal of Applied Mathematics
Volume 39 No. 1s, 2026
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Signature evasion techniques, including polymorphic attacks and protocol manipulation, accounted for 42.6% of detection failures. These counterexamples demonstrated that attackers could achieve their objectives through minor variations of known attack patterns that fell outside the strict matching criteria of signature databases. False positives from legitimate traffic patterns constituted 60.5% of failures, primarily affecting the anomaly-based system. These included legitimate administrative activities, bulk data transfers, and traffic spikes during normal operational periods that exceeded statistical thresholds.

The signature-based IDS exhibited the smallest state space and fastest verification time due to its deterministic rule-matching logic. The anomaly-based and hybrid systems required significantly more computational resources due to probabilistic state transitions and the need to model statistical distributions over normal behaviour.

The hybrid IDS achieved full verification of all seven properties after three refinement iterations. Key improvements included implementing a weighted voting mechanism between signature and anomaly components, adding temporal correlation analysis to reduce false positives, and introducing adaptive thresholds that adjust based on network context. The signature-based IDS plateaued at 6/7 properties due to the fundamental limitation that finite signature databases cannot achieve completeness against evolving attack landscapes. The anomaly-based IDS improved to 5/7 properties but continued to exhibit trade-offs between false positive and false negative rates that prevented simultaneous satisfaction of completeness and soundness.

Verification time exhibited approximately polynomial growth ($O(n^{2.5})$) with respect to the number of detection rules, while memory consumption scaled more steeply ($O(n^{3.2})$). These results indicate that formal verification remains tractable for moderate-complexity IDS implementations but faces challenges when scaling to enterprise-grade systems with thousands of detection rules.

Overall, the formal verification process uncovered 17 previously unknown vulnerabilities across the three IDS implementations. These included race conditions in concurrent rule evaluation (3 instances), incomplete state transitions allowing attack sequences to bypass detection (8 instances), and logical errors in threshold calculations (6 instances). Each vulnerability was confirmed through manual code inspection and subsequently patched, demonstrating the practical value of formal verification in identifying subtle implementation flaws. These vulnerability findings are supported by the literature.

References

1. Alpcan, T., & Basar, T. (2003). A game theoretic approach to decision and analysis in network intrusion detection. *42nd IEEE international conference on decision and control (IEEE Cat. No. 03CH37475), 9-12 December 2003, Maui, HI, USA. 3*, pp. 2595-2600. IEEE. doi:https://doi.org/10.1109/CDC.2003.1273013
2. Avalle, M., Pironti, A., & Sisto, R. (2014). Formal verification of security protocol implementations: a survey. *Formal Aspects of Computing, 26*(1), 99-123. doi:https://doi.org/10.1007/s00165-012-0269-9

Received: Aug 08, 2025

Volume 39 No. 1s, 2026
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

3. Cárdenas, A. A., Baras, J. S., & Seamon, K. (2006). A framework for the evaluation of intrusion detection systems. *2006 IEEE Symposium on Security and Privacy (S&P'06), 21-24 May 2006, Berkeley/Oakland, CA, USA* (p. 15 pp). IEEE. doi:https://doi.org/10.1109/SP.2006.2

4. Chen, A., Kwon, Y., & Lee, K. H. (2025). Racedb: Detecting Request Race Vulnerabilities in Database-Backed Web Applications. *2025 IEEE Symposium on Security and Privacy (SP), 12-15 May 2025, San Francisco, CA, USA* (pp. 939-955). IEEE. doi:https://doi.org/10.1109/SP61157.2025.00029

5. Chen, H., Dean, D., & Wagner, D. A. (2004). Model Checking One Million Lines of C Code. *NDSS*, *4*, pp. 171-185. Retrieved December 31, 2025, from https://www.cs.ucdavis.edu/~hchen/paper/ndss2004.pdf

6. Debar, H., Dacier, M., & Wespi, A. (2000). A revised taxonomy for intrusion-detection systems. *Annales des télécommunications, 55*(7), 361-378. doi:https://doi.org/10.1007/BF02994844

7. Engler, D., & Ashcraft, K. (2003). RacerX: Effective, static detection of race conditions and deadlocks. *ACM SIGOPS operating systems review, 37*(5), 237-252. doi:https://doi.org/10.1145/1165389.945468

8. Faizal, M. A., Zaki, M. M., Shahrin, S., Robiah, Y., Rahayu, S. S., & Nazrulazhar, B. (2009). Threshold verification technique for network intrusion detection system. *arXiv, 0906*, 3843. doi:https://doi.org/10.48550/arXiv.0906.3843

9. Flood, R., Casadio, M., Aspinall, D., & Komendantskaya, E. (2025). Formally Verifying Robustness and Generalisation of Network Intrusion Detection Models. *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing* (pp. 1867-1876). ACM. doi:https://doi.org/10.1145/3672608.3707927

10. Fogla, P., & Lee, W. (2006). Evading network anomaly detection systems: formal reasoning and practical techniques. *13th ACM conference on Computer and communications security* (pp. 59-68). ACM. doi:h□ps://doi.org/10.1145/1180405.1180414

11. Goseva-Popstojanova, K., Wang, F., Wang, R., Gong, F., Vaidyanathan, K., Trivedi, K., & Muthusamy, B. (2001). Characterising intrusion tolerant systems using a state transition model. *Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01, 12-14 June 2001, Anaheim, CA, USA. 2*, pp. 211-221. IEEE. doi:https://doi.org/10.1109/DISCEX.2001.932173

12. Gu, G., Fogla, P., Dagon, D., Lee, W., & Skoric, B. (2006). Towards an information-theoretic framework for analyzing intrusion detection systems. In D. Gollmann, J. Meier, & A. Sabelfeld (Ed.), *European symposium on research in computer security. Lecture Notes in Computer Science, vol 4189*, pp. 527-546. Springer Berlin, Heidelberg. doi:https://doi.org/10.1007/11863908_32

Received: Aug 08, 2025

Volume 39 No. 1s, 2026
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

13. Gu, G., Fogla, P., Dagon, D., Lee, W., & Skoric, B. (2006). Towards an information-theoretic framework for analyzing intrusion detection systems. In D. Gollmann, J. Meier, & A. Sabelfeld (Ed.), *European symposium on research in computer security, March 21-24, 2006, Taipei, Taiwan. Lecture Notes in Computer Science, vol 4189*, pp. 527-546. Springer Berlin, Heidelberg. doi:https://doi.org/10.1007/11863908_32

14. Hailesellasie, M., & Hasan, S. R. (2018). Intrusion detection in PLC-based industrial control systems using formal verification approach in conjunction with graphs. *Journal of Hardware and Systems Security, 2*(1), 1-14. doi:https://doi.org/10.1007/s41635-017-0017-y

15. Ilgun, K., Kemmerer, R. A., & Porras, P. A. (2002). State transition analysis: A rule-based intrusion detection approach. *IEEE transactions on software engineering, 21*(3), 181-199. doi:https://doi.org/10.1109/32.372146

16. Ko, C., & Redmond, T. (2002). Noninterference and intrusion detection. *Proceedings 2002 IEEE Symposium on Security and Privacy, 12-15 May 2002, Berkeley, CA, USA* (pp. 177-187). IEEE. doi:https://doi.org/10.1109/SECPRI.2002.1004370

17. Krichen, M. (2023). Formal methods and validation techniques for ensuring automotive systems security. *Information, 14*(12), 666. doi:https://doi.org/10.3390/info14120666

18. Li, X., & Xue, Y. (2011). Block: a black-box approach for detection of state violation attacks towards web applications. *Proceedings of the 27th Annual Computer Security Applications Conference* (pp. 247-256). ACM. doi:https://doi.org/10.1145/2076732.2076767

19. Rouached, M., & Sallay, H. (2012). An efficient formal framework for intrusion detection systems. *Procedia Computer Science, 10*, 968-975. doi:https://doi.org/10.1016/j.procs.2012.06.132

20. Sánchez, C., Schneider, G., Ahrendt, W., Bartocci, E., Bianculli, D., Colombo, C., & Falcone, Y. e. (2019). A survey of challenges for runtime verification from advanced application domains (beyond software). *Formal Methods in System Design, 54*(3), 279-335. doi:https://doi.org/10.1007/s10703-019-00337-w

21. Upadhyay, A., Laxmi, V., & Naval, S. (2023). Navigating the Concurrency Landscape: A Survey of Race Condition Vulnerability Detectors. *arXiv, 2312*, 14479. doi:https://doi.org/10.48550/arXiv.2312.14479

22. Veronica, S. (2025). Reasoning Under Threat: Symbolic and Neural Techniques for Cybersecurity Verification. *arXiv, 2503*, 22755. doi:https://doi.org/10.48550/arXiv.2503.22755

23. Verwoerd, T., & Hunt, R. (2002). Intrusion detection techniques and approaches. *Computer communications, 25*(15), 1356-1365. doi:https://doi.org/10.1016/S0140-3664(02)00037-3

Received: Aug 08, 2025

Volume 39 No. 1s, 2026
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

24. Yadav, P., & Gaur, M. (2018). A behavioural theory for intrusion detection system in mobile ad-hoc networks. *Proceedings of the 2nd International Conference on High Performance Compilation, Computing and Communications* (pp. 51-60). ACM. doi:https://doi.org/10.1145/3195612.3195617

Received: Aug 08, 2025