КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 2 (34), 2026
ISSN 2663 – 4023
DOI 10.28925/2663-4023.2026.34.1230
УДК 004.056.5:004.8:004.738.5

Kostiantyn Savchuk
Postgraduate Student of the Department of Information Technology Security
Lviv Polytechnic National University, Lviv, Ukraine
ORCID:0009-0006-6612-979X
kostiantyn.v.savchuk@lpnu.ua

Elena Nyemkova
Doctor of Technical Sciences, Professor,
Professor of the Department of Information Technology Security
Lviv Polytechnic National University, Lviv, Ukraine
ORCID:0000-0003-0690-2657
olena.a.niemkova@lpnu.ua

TWO-LAYER WEB APPLICATION SECURITY MODEL BASED ON OWASP TOP 10:2025 AND ARTIFICIAL INTELLIGENCE METHODS

Abstract. Web applications continue to be the primary target for cyberattacks such as SQL injection (SQLi), cross-site scripting (XSS), cross-site request forgery (CSRF), and distributed denial-of-service (DDoS). The traditional methods of defense – signature matching, static rule sets, and perimeter control – can no longer keep up with the dynamic threats that modern cloud-based and API-driven systems face on a daily basis. In this paper, we propose a two-layer security model framework designed to bridge this gap. The framework is organized into two complementary layers. The first layer uses deterministic security controls aligned with the OWASP Top 10:2025 standard, providing a stable and auditable baseline of protection. The second layer integrates three complementary AI techniques: HTTP request embeddings (HTTP2vec) for semantic representation of traffic, Graph Neural Networks (GNNs) for modeling relational patterns between users, sessions, and endpoints, and an autoencoder-based online anomaly detector called Kitsune, which adapts to evolving traffic in real time. We use a structured analytical approach in this study. First, we review recent research papers and real-world incident reports from industry, and then we systematically compare different detection methods across accuracy, latency, and operational cost. The results show that traditional controls remain effective against known attack signatures, while AI-driven methods are far better at uncovering zero-day exploits and coordinated, low-and-slow campaigns that rule-based systems typically miss. Industry reports indicate that integrating AI into SIEM and SOAR platforms can reduce false-positive alerts by approximately 60% and accelerate incident investigation by about 40%, freeing analysts to focus on higher-value tasks. We also argue that Precision–Recall curves and the Numenta Anomaly Benchmark (NAB) are more appropriate evaluation tools than ROC-AUC for security data, where malicious traffic often represents less than 0.1% of total volume and class imbalance distorts conventional metrics. The main conclusion is simple but important: AI performs best not as a replacement for traditional controls, but as a force multiplier that strengthens them, producing a layered defense that is both resilient and adaptive.

Keywords: web application security; OWASP Top 10; artificial intelligence; anomaly detection; graph neural networks; intrusion detection; two-layer security model; cyber threat protection.

INTRODUCTION

Today, more than 5.5 billion people around the world use the Internet [1]. Web applications are at the centre of almost every digital service – from online shopping and banking to education and government portals. This wide use also makes web applications an attractive target for attackers. The most common attacks include SQL injection (SQLi), cross-site scripting (XSS), cross-site request forgery (CSRF), and distributed denial-of-service (DDoS) [2]. Year after year, the Verizon Data Breach Investigations Report lists web application attacks among the top causes of confirmed data breaches [3].

One reason detection is so difficult is the sheer volume of normal traffic. A recent study of Internet traffic categories shows that video streaming platforms like YouTube produce about 1,500 terabytes of data each day;

social media platforms generate around 900 TB daily; and news sites together with e-commerce websites add up to roughly 650 TB per day [4]. Attackers take advantage of this. They hide malicious requests inside the massive flow of legitimate data, which makes it hard for security tools to tell the difference.

The older methods of web security were built for a different era. Perimeter firewalls, static rule sets, and signature-based detection worked reasonably well when websites were simple and mostly static. But today's applications run on microservices, communicate through APIs, and are deployed across multiple cloud providers [5]. A signature-based Web Application Firewall (WAF) can only stop attacks it already knows. If the attack is new or slightly modified, the WAF will let it pass [12, 13].

The numbers paint a worrying picture. According to industry reports, 36% of all cyberattacks worldwide involve phishing. Human errors are responsible for 82% of cybersecurity incidents, and a single data breach costs an average of $4.45 million [4, 6]. Security teams in many organizations are overwhelmed. Their monitoring tools generate thousands of alerts every day, and a large share of those alerts turn out to be false positives. This leads to what practitioners call "alert fatigue" – analysts become desensitised and may miss real threats hidden among the noise [7].

All of this points to a clear gap. We need an approach that keeps the reliable, predictable protection of traditional controls while adding the ability to detect new and unexpected attacks. Several AI-based techniques for web security have been studied individually [8, 9, 10], but there has been limited effort to bring them together in a structured way and connect them to an established security framework like the OWASP Top 10.

In this paper, we address this gap. We propose a two-layer framework. The bottom layer – which we call the Foundation – consists of deterministic controls that map directly to the OWASP Top 10:2025, the latest edition of the OWASP risk taxonomy [2]. The top layer – the Amplifier – uses three AI techniques chosen because they cover different types of threats. HTTP2vec [8] detects anomalous HTTP requests by learning the patterns of normal traffic. Graph Neural Networks [9] find coordinated attacks by modelling relationships between users, devices, and sessions. Kitsune [10] provides real-time anomaly detection at the network edge using lightweight autoencoders.

Our contributions are fourfold. First, we present a two-layer framework that maps specific AI techniques to specific OWASP Top 10:2025 risk categories. Second, we compare traditional and AI-based detection across multiple criteria. Third, we collect and discuss industry evidence about the benefits of adding AI to SIEM and SOAR platforms. Fourth, we recommend evaluation metrics that are better suited to the extreme class imbalance of real-world security data.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 describes the research problem. Section 4 explains the methodology used in this study. Section 5 presents the proposed framework. Section 6 shows the results of the comparison. Section 7 discusses the main findings and the limitations. Section 8 provides the conclusion and ideas for future research.

Related Work. The common tools used to protect web applications include Web Application Firewalls (WAFs), Intrusion Detection Systems (IDS), and Intrusion Prevention Systems (IPS). WAFs check incoming HTTP requests and block those that match known attack patterns [11, 13]. IDS and IPS systems monitor network traffic and look for signs of known attacks or exploits [12]. On top of this, developers apply input validation, access control, and encryption.

These tools do their job well when the threat is known and well-documented. The problem starts when the threat is new. Signature-based systems cannot detect an attack they have never seen before [17, 22]. Rule-based WAFs also tend to produce a lot of false alarms, especially when the application accepts complex user input [13]. Another issue is the assumption behind perimeter security: that there is a clear line between the "trusted" internal network and the "untrusted" outside. In cloud-native environments and zero-trust architectures, this line no longer exists [5].

Researchers have tried many AI methods to solve the problems that traditional tools cannot handle. We can group them into three main areas.

NLP-based methods for HTTP traffic analysis. Gniewkowski et al. [8] developed HTTP2vec, a system that treats HTTP requests as text sequences and uses a RoBERTa language model to learn their structure. The model trains only on normal traffic. When a new request arrives, the system converts it into a vector and checks how far it is from the normal patterns. If the distance is large, the request is likely malicious. The authors tested this on three datasets (CSIC2010, CSE-CIC-IDS2018, and a custom dataset) and showed that the approach can detect SQL injection and command injection attacks without needing labelled attack examples. Park et al. [15] took a different route: they converted HTTP messages into binary images and used convolutional autoencoders to spot anomalies. Li et al. [24] explored weighted word2vec for similar purposes.

Graph-based methods. Bilot et al. [9] published a broad survey of Graph Neural Networks (GNNs) for intrusion detection. The idea is to model users, devices, sessions, and IP addresses as nodes in a graph and their

interactions as edges. This makes it possible to see patterns that sequential log analysis would miss – for example, a chain of compromised accounts used by the same attacker, or a coordinated brute-force attack coming from many IPs at once. The survey reports promising results but also notes challenges with scalability and the lack of publicly available graph-structured security datasets.

Online anomaly detection. Mirsky et al. [10] created Kitsune, a system that uses an ensemble of small autoencoders deployed at the network edge. Each autoencoder learns how a subset of traffic features normally behaves. If incoming traffic causes high reconstruction error, Kitsune flags it as an anomaly. Because it learns from unlabelled data and runs on low-power hardware, Kitsune is well suited for gateways and IoT devices. Tests showed it can detect DDoS, reconnaissance scans, and man-in-the-middle attacks effectively. Vartouni et al. [25] explored a similar concept with stacked autoencoders. Table 1 compares these three approaches side by side.

Table 1

Comparison of AI-based web attack detection approaches

Characteristic	HTTP2vec [8]	GNNs [9]	Kitsune [10]
Core technique	NLP embeddings (RoBERTa)	Graph neural networks	Autoencoder ensembles
Learning type	Unsupervised	Supervised / semi-supervised	Unsupervised
Input data	Raw HTTP requests	Graph-structured logs	Network packet features
What it detects	SQLi, XSS, command injection	Coordinated attacks, account takeover	DDoS, reconnaissance, MitM
Speed	Moderate (batch)	Moderate to high	Low latency (real-time)
Needs labelled data?	No	Usually yes	No
Where it runs	Application layer	Analytics platform	Network edge / gateway
Main weakness	Large model size (~3000 dim.)	Hard to scale on big graphs	Only sees network-level features

The OWASP Top 10 is a well-known standard that lists the most critical web application security risks [2]. It is updated every few years based on data collected from the security community. Previous editions were released in 2013, 2017, and 2021 [28].

The newest edition – OWASP Top 10:2025 – came out in November 2025 [2, 14]. It brings two new categories: Software Supply Chain Failures (A03:2025) and Mishandling of Exceptional Conditions (A10:2025). Server-Side Request Forgery (SSRF), which had its own category in 2021, is now part of Broken Access Control (A01:2025). The 2025 edition also tries to focus more on root causes rather than symptoms – for example, it names "Cryptographic Failures" instead of "Sensitive Data Exposure" [14]. Table 2 lists all ten categories with notes on what changed.

Table 2

OWASP Top 10:2025 risk categories

Rank	Category	Changes from 2021
A01	Broken Access Control	Now includes SSRF
A02	Security Misconfiguration	Moved up from #5
A03	Software Supply Chain Failures	New (expanded from Vulnerable Components)
A04	Cryptographic Failures	Dropped from #2
A05	Injection	Dropped from #3
A06	Insecure Design	Dropped from #4
A07	Identification and Authentication Failures	Renamed
A08	Data Integrity Failures	Stable
A09	Security Logging and Alerting Failures	More focus on actionable alerts
A10	Mishandling of Exceptional Conditions	New

After reviewing the literature, we identified three key gaps. First, most studies focus on a single detection method. There is limited research on how to combine AI methods with traditional security controls into one

structured system. Second, the OWASP Top 10 is widely used for risk assessment, but nobody has systematically used it as a mapping tool to decide which AI technique should cover which risk. Third, many researchers evaluate their detection models using ROC-AUC, which can be misleading when malicious traffic is extremely rare compared to normal traffic [16, 18]. These gaps are what our study tries to fill.

Problem Statement. The central question of this paper is: how should an organisation combine traditional web security controls with AI-driven detection to get a defense that is both reliable and adaptive?

Neither approach works well alone. Traditional controls are predictable and easy to understand, but they cannot catch attacks they have never seen. AI models can detect new patterns, but they are sensitive to adversarial tricks, they suffer from concept drift as traffic patterns change over time, and they cannot offer the kind of hard guarantees that compliance requirements demand [17, 22].

The aim of this work is to develop and substantiate a concept for integrating traditional (deterministic) web security controls with artificial intelligence methods within a unified two-layer framework to ensure robust and adaptive protection against modern threats.

To achieve this aim, the study addresses the following research questions:

– How can deterministic controls based on the OWASP Top 10:2025 be combined with AI-driven detection within this two-layer framework?

– Which AI techniques work best for specific types of web threats?

– What evaluation metrics should be used when the data is extremely imbalanced – specifically, when less than 0.1% of all traffic is malicious?

RESEARCH METHODOLOGY

We used a systematic analytical approach. This means we did not run our own experiments or collect original data. Instead, we reviewed published research and real-world deployment reports, and then compared the different approaches based on clear criteria. Figure 1 shows the seven stages of our method.

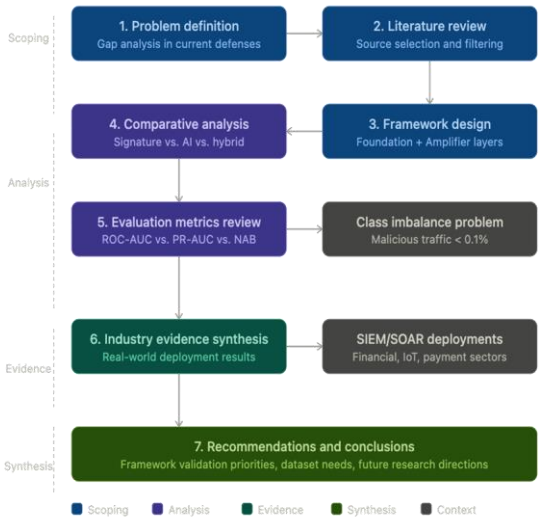

Fig. 1. Research methodology workflow

The study proceeds through seven stages grouped into four phases: scoping (problem definition, literature review, framework design), analysis (comparative evaluation of detection approaches, metrics review), evidence (industry deployment synthesis), and synthesis (recommendations and conclusions).

How we selected sources. We searched IEEE Xplore, the ACM Digital Library, Springer, and arXiv for papers published between 2018 and 2025. The search used keywords such as “web application security,” “anomaly detection,” “HTTP embeddings,” “graph neural networks,” “OWASP,” “intrusion detection,” and “SIEM”. We also included industry reports when they described specific deployments with measurable results, but we always distinguish these from peer-reviewed work.

What we compared. We evaluated detection approaches on six criteria: (1) how accurately they detect attacks, (2) how many false alarms they produce, (3) how fast they make decisions, (4) how well they handle growing traffic volumes, (5) whether they need labelled training data, and (6) how explainable their decisions are.

What this methodology cannot do. Because we did not run experiments, our proposed framework remains a theoretical design. The evidence we present comes from other researchers' experiments and from industry reports. Our conclusions should be read with this in mind.

RESEARCH RESULTS

Architecture Overview. Our framework has two layers that work together, connected through SIEM and SOAR platforms (Figure 2).

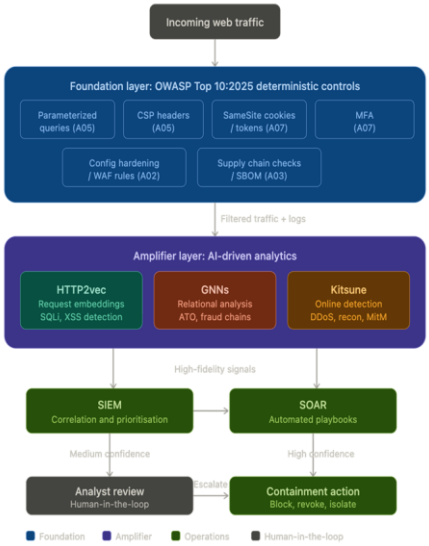

Fig. 2. Architecture of the proposed two-layer framework

The Foundation layer implements deterministic controls mapped to OWASP Top 10:2025 categories. The Amplifier layer deploys three AI techniques – HTTP2vec for application-level anomaly detection, GNNs for relational threat analysis, and Kitsune for real-time network-edge detection. Both layers feed signals into SIEM for correlation and SOAR for automated response. Medium-confidence alerts are routed to a human analyst; high-confidence alerts trigger automated containment.

The design is based on a simple idea: strengthen, not replace. AI does not replace traditional controls, it helps make them better. The foundation layer deals with known threats using rules that work in a consistent way. The amplifier layer helps detect new threats that rules cannot catch. The SIEM system collects and connects all signals, and the SOAR system can automatically respond when the alert is reliable.

Foundation Layer: Deterministic Controls. This layer implements controls that directly address the risks listed in the OWASP Top 10:2025 [2]. Each control is well-established and has a clear, deterministic effect.

Parameterized queries and prepared statements. These prevent SQL injection by keeping user input separate from SQL code. When implemented correctly, they eliminate the SQLi attack vector entirely. This addresses A05:2025 (Injection), which dropped to fifth place in the 2025 ranking but remains one of the most exploited vulnerabilities in practice [2].

Content Security Policy (CSP). CSP headers tell the browser which sources are allowed to load scripts, styles, and other resources. This limits the damage an XSS attack can do, even if the attacker manages to inject a script. CSP works as a defence-in-depth measure alongside server-side input validation. It helps address XSS risks under A05:2025.

SameSite cookies and short-lived tokens. The SameSite attribute stops the browser from sending cookies along with cross-site requests, which prevents many CSRF attacks. Short-lived access and refresh tokens reduce the time window an attacker has to exploit a stolen token. These controls address session-related risks in A07:2025 (Identification and Authentication Failures).

Multi-factor authentication (MFA). Adding a second verification step – such as a one-time code or biometric check – makes credential theft much less useful to attackers. MFA is especially important given the high rate of account takeover (ATO) attacks reported across industries [6, 7]. This addresses A07:2025.

Security configuration hardening. Security Misconfiguration is now ranked as A02:2025, which makes it the second most serious risk in the new list. Hardening means removing default passwords, disabling services that are not needed, setting correct security headers, and checking that configurations are secure in all

environments. Many breaches happen not because of clever exploits, but because someone left a default password or an open cloud storage bucket [2, 14].

Supply chain integrity checks. A03:2025 (Software Supply Chain Failures) is a new category that reflects how often attackers now target dependencies, build tools, and distribution systems. Controls include scanning dependencies for known vulnerabilities, maintaining a Software Bill of Materials (SBOM), and verifying the integrity of the build pipeline [2].

Protection of infocommunication systems also depends on understanding the types of traffic they carry – real-time, streaming, elastic, and signalling – each with its own protocols and specific vulnerabilities [4, 26]. The deterministic controls described above must be adapted to match the particular traffic profile of each system.

Other marks. The three AI techniques in this layer were chosen because they cover different types of threats at different points in the architecture.

HTTP Request Embeddings (HTTP2vec). HTTP requests have a regular structure, much like sentences in a language. HTTP2vec [8] takes advantage of this. It uses a RoBERTa-based language model to learn what "normal" HTTP requests look like by training only on legitimate traffic. The model converts each request into a dense numerical vector. During operation, incoming requests are converted in the same way. If a request's vector is far from the normal cluster, the system flags it as suspicious.

This approach is good at catching injection attacks – SQLi, XSS, command injection – because malicious payloads break the learned patterns of normal traffic in distinctive ways. The main strength is that it needs no labelled attack data for training, which matters because labelled datasets in web security are scarce and go out of date quickly. The main drawback is the model's size: the vector space has about 3,000 dimensions, which means training takes time and needs significant computing power [8].

Graph Neural Networks (GNNs). Sequential log analysis processes events one at a time or in time order. It can miss attacks that involve multiple actors working together. GNNs [9] take a different approach: they build a graph where users, devices, sessions, and IP addresses are nodes, and their interactions are edges.

This graph structure makes certain attack patterns visible. For example, if an attacker takes control of one account and then uses it to access others, GNNs can detect the unusual connections between them. The same idea applies to brute-force attacks coming from many IP addresses or fraud cases where several accounts act together. These patterns are linked to A01:2025 (Broken Access Control) and A07:2025 (Authentication Failures). However, GNNs need data in a graph format, which is not always easy to get, and they may have problems when working with very large graphs [9].

Online Anomaly Detection (Kitsune). Kitsune [10] is designed for speed and simplicity. It runs an ensemble of small autoencoders right at the network gateway or on edge devices. Each autoencoder learns to reconstruct a subset of network traffic features. When new traffic produces a high reconstruction error, the system calls it anomalous.

Because the autoencoders train themselves on whatever traffic they see – no labels needed – Kitsune adapts to the specific environment where it is deployed. It is good at catching DDoS attacks, port scanning, and man-in-the-middle attacks, all of which create statistical signatures that differ clearly from normal traffic. Its lightweight design makes it suitable even for IoT devices with limited processing power [10]. Table 3 shows how these three techniques map to the OWASP Top 10:2025 categories.

Table 3

Mapping of AI techniques to OWASP Top 10:2025 risk categories

OWASP Top 10:2025 Category	HTTP2vec	GNNs	Kitsune	Main Deterministic Control
A01: Broken Access Control	–	✓	–	RBAC, least privilege
A02: Security Misconfiguration	–	–	✓	Configuration hardening
A03: Supply Chain Failures	–	✓	–	SBOM, dependency scanning
A05: Injection	✓	–	–	Parameterized queries, CSP
A07: Authentication Failures	–	✓	–	MFA, SameSite cookies
A09: Logging and Alerting	✓	✓	✓	Centralized logging
Network-level (DDoS, recon)	–	–	✓	Rate limiting, WAF rules

Online Anomaly Detection (Kitsune). An AI model that generates alerts but is not connected to anything useful does not help much. The real value comes from integration with the tools that security teams already use. Figure 3 shows the data flow.

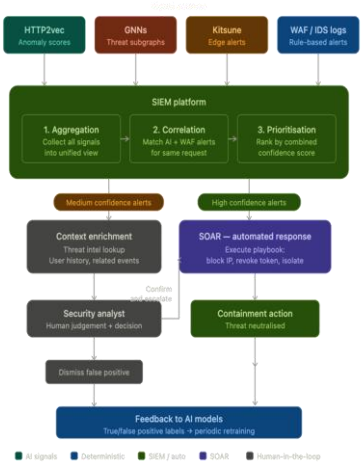

Fig. 3. Data flows between SIEM and SOAR systems

Four main sources send signals: HTTP2vec anomaly scores, GNN results that show suspicious connections, Kitsune alerts from the network edge, and logs from traditional tools like WAF and IDS. All these signals go into the SIEM system, where they are collected, combined, and prioritised.

If an alert has high confidence, the SOAR system can automatically run response actions to stop the threat. Alerts with medium confidence are enriched with extra information, such as threat intelligence and user data, and then sent to a security analyst for review. The decisions made by analysts, including confirmed threats and false positives, are later used to improve and retrain the AI models.

The SIEM platform does three things in this architecture. It collects signals from both layers and puts them in one place. It correlates AI-generated anomaly scores with deterministic alerts – for instance, if HTTP2vec flags a request and the WAF also triggers on it, the combined confidence is much higher than either signal alone. And it filters and ranks alerts so that analysts see the most important ones first.

The SOAR platform automates the response when confidence is high enough. Automated playbooks can block IP addresses, revoke tokens, or quarantine affected systems without waiting for a human. For cases where the system is not sure, SOAR adds extra context – threat intelligence lookups, user history, related events – and passes the enriched alert to an analyst.

This design directly addresses the alert fatigue problem from Section 1. By combining signals from traditional controls with signals from AI and then filtering them, the system reduces the number of false alerts that reach human analysts.

Comparative analysis and evaluation. Table 4 compares three approaches: pure signature-based, pure AI-based, and the two-layer model we propose.

Table 4

Multi-criteria comparison of detection paradigms

Criterion	Signature-based	AI-based (anomaly)	Two-layer (proposed)
Known threats	High detection	Moderate detection	High detection
Novel threats	Low detection	High detection	High detection
False positive rate	Moderate to high	Depends on model	Reduced (~60% lower)
Speed	Fast	Varies (batch to real-time)	Fast to moderate
Scalability	Good	Depends on model	Moderate to good
Needs labelled data	Yes (signature updates)	Low to moderate	Low
Explainability	High (rules are readable)	Low to moderate	Moderate (mixed signals)
Maintenance	High (constant rule updates)	Moderate (periodic retraining)	Moderate
Evasion resistance	Low (bypass techniques exist)	Moderate (adversarial risk)	Higher (two layers to beat)

The two-layer approach performs better on most criteria because it uses the strengths of both methods. Rule-based controls give stable and clear protection against known attacks. AI adds the ability to detect threats

that rules may miss. Also, since an attacker would need to bypass both layers at the same time, the system is harder to evade than using only one method.

Evidence from Industry Deployments. Several organizations have shared results after adding AI to their security operations. We collected these numbers from published reports, but it is important to note their limits. These are not results from controlled experiments. Each organization had its own traffic, tools, and types of attacks. So, these numbers should be seen as general indicators, not exact guarantees. Table 5 summarizes what has been reported [7].

Table 5

Reported outcomes of AI integration in security operations across organisations

What was measured	Result	Where
False positive alerts	~60% fewer	SIEM/SOAR platforms
Investigation time	~40% faster	SOC operations
Account takeover incidents	65% fewer per year	Financial institutions
Fraudulent transactions blocked	Over 80 million	Payment platforms
True positive detections	From 11 to 73	One financial firm
Detection accuracy	96-97%	Smart city IoT systems
Response time	Under 30 seconds	Smart city IoT systems

A note on these figures. All values in Table 5 come from a single industry aggregation source [7], which compiles case studies from multiple companies. We could not confirm every number by checking its original source. Because of this, these numbers should be seen as general indicators, not exact values. However, all cases show the same trend: using AI leads to fewer false alerts, faster response, and better detection. Future work should try to find and verify the original studies behind these results.

Why Standard Metrics Can Be Misleading. Choosing the right evaluation metric matters more than most people realize, especially in security. The root of the problem is class imbalance. In a typical network, malicious traffic is a tiny fraction of the total – often less than 0.1% [16]. This creates problems for commonly used metrics.

The issue with ROC-AUC. The Receiver Operating Characteristic curve plots true positive rate against false positive rate. When the negative class (normal traffic) is huge, even a classifier that generates many false alarms in absolute terms can have a very low false positive rate as a percentage. The result is a high ROC-AUC score that looks great on paper but hides the fact that the analyst's screen is full of junk alerts [16, 18].

Why Precision-Recall is better here. The Precision-Recall (PR) curve plots precision (what fraction of flagged items are actually malicious) against recall (what fraction of malicious items were caught). This directly reflects what the security team experiences: the ratio of useful alerts to noise. The Average Precision (AP) score summarizes the PR curve as a single number. Figure 4 shows a conceptual comparison.

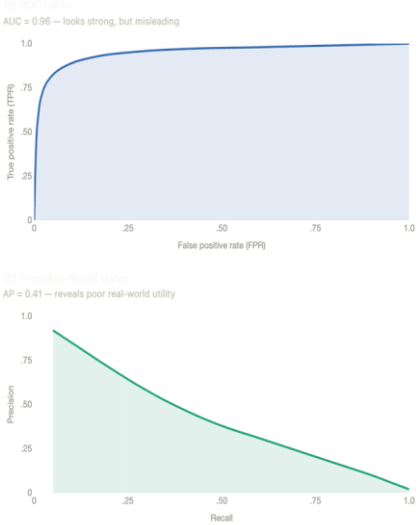

Fig. 4. Comparison of ROC and Precision-Recall curves for the same classifier on class-imbalanced security data (99.9% benign, 0.1% malicious). (a) The ROC curve suggests strong performance with an AUC of 0.96. (b) The Precision-Recall curve reveals that at 70% recall, only 24% of flagged alerts are true positives

For web security applications where malicious traffic is a very small fraction of total volume, Precision-Recall curves provide a more accurate picture of practical classifier utility than ROC-AUC.

For real-time systems: the Numenta Anomaly Benchmark. When detection speed matters – as it does for Kitsune and similar edge-deployed systems – the Numenta Anomaly Benchmark (NAB) [19] provides a better evaluation framework. NAB gives higher scores to systems that detect anomalies early and penalizes late detections. This matches real security needs: an alert that appears five minutes after an attack starts is much less useful than one that comes in the first few seconds.

We suggest that researchers report Precision-Recall curves and AP scores together with ROC-AUC, and that systems working in real time be evaluated using the NAB method.

CONCLUSIONS AND FUTURE RESEARCH

Our analysis supports the idea that AI works best as an amplifier, not a replacement. The three techniques we selected – HTTP2vec, GNNs, and Kitsune – address different types of threats at different architectural layers. HTTP2vec works at the application level on individual requests. GNNs work at the analytics level on relationships between entities. Kitsune works at the network edge on raw traffic statistics. This spread is deliberate. A single AI technique, no matter how good, would leave gaps.

The mapping to OWASP Top 10:2025 categories (Table 3) is a practical tool. Instead of asking "should we invest in AI for security?" – a question that is too broad to answer – organizations can ask "which of our OWASP risks would benefit most from AI support?" and then choose accordingly.

The new categories in the 2025 edition are relevant to our framework. Software Supply Chain Failures (A03:2025) reflects the growing threat from compromised dependencies, build systems, and distribution channels. This is an area where GNNs could prove useful, since supply chain attacks often involve chains of trust relationships that look normal individually but are suspicious as a pattern. Mishandling of Exceptional Conditions (A10:2025) highlights the risk of applications failing in unsafe ways. AI-based monitoring, especially at the application level, can detect unusual system states that might indicate exploitation or imminent failure – for example, a sudden spike in error responses from a specific endpoint. The merger of SSRF into Broken Access Control (A01:2025) recognizes that in microservice architectures, the boundary between internal and external access is blurring. GNNs, which model relationships between services and users, are well positioned to detect SSRF attacks that exploit trust between internal services [14].

Research on semiotic models for cyberspace protection [26] offers a complementary perspective. By analyzing traffic at semantic, syntactic, and pragmatic levels, such models can capture not just the technical characteristics but also the social context of the data being transmitted, which is particularly relevant for detecting social engineering attacks [4, 27].

Based on our analysis, we have a few recommendations for practitioners. Do not skip the fundamentals. Make sure your OWASP Top 10:2025 controls are properly implemented before spending money on AI. AI cannot fix a missing parameterized query or a default admin password.

Choose AI techniques based on your risk profile. If injection attacks are your main concern, start with HTTP2vec. If you deal with complex user interactions and worry about account takeover, look at GNNs. If network-level threats like DDoS are the priority, Kitsune or a similar edge-deployed system makes sense.

Integrate through SIEM and SOAR. AI signals that sit in a separate dashboard and nobody looks at are a waste of resources. Feed them into the tools your team already uses, correlate them with deterministic alerts, and automate the obvious responses.

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. International Telecommunication Union. (2024). *Facts and figures 2024: Internet use*. ITU. https://www.itu.int/itu-d/reports/statistics/2024/11/10/ff24-internet-use/
2. OWASP Foundation. (2025). *OWASP Top 10:2025*. https://owasp.org/Top10/2025/
3. Verizon Business. (2024). *2024 data breach investigations report*. https://www.verizon.com/business/resources/reports/dbir.html
4. Tolkachov, M. Yu. (2024). Mechanisms of traffic protection in cyberspace. *Suchasnyi Zakhyst Informatsii*, *4*(60), 85-99. https://doi.org/10.31673/2409-7292.2024.040009
5. Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020). *Zero trust architecture* (NIST Special Publication 800-207). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-207

6. IBM Security. (2023). *Cost of a data breach report 2023*. IBM. https://d110erj175o600.cloudfront.net/wp-content/uploads/2023/07/25111651/Cost-of-a-Data-Breach-Report-2023.pdf
7. Dilmegani, C. (2025). *Top 13 AI cybersecurity use cases with real examples in 2025*. AIMultiple Research. https://research.aimultiple.com/ai-cybersecurity-use-cases/
8. Gniewkowski, M., Maciejewski, H., Surmacz, T., & Walentynowicz, W. (2021). *HTTP2vec: Embedding of HTTP requests for detection of anomalous traffic* (arXiv:2108.01763). arXiv. https://doi.org/10.48550/arXiv.2108.01763
9. Bilot, T., Legay, A., & Rønne, P. B. (2023). Graph neural networks for intrusion detection: A survey. *IEEE Access, 11*, 45589-45612. https://doi.org/10.1109/ACCESS.2023.3275789
10. Mirsky, Y., Doitshman, T., Elovici, Y., & Shabtai, A. (2018). Kitsune: An ensemble of autoencoders for online network intrusion detection. In *Proceedings of the Network and Distributed System Security Symposium (NDSS 2018)*. Internet Society. https://doi.org/10.14722/ndss.2018.23204
11. Razzaq, A., Hur, A., Ahmad, H. F., & Masood, M. (2013). Cyber security: Threats, reasons, challenges, methodologies and state-of-the-art solutions for industrial applications. In *2013 IEEE 11th International Symposium on Autonomous Decentralized Systems (ISADS)* (pp. 1–6). IEEE. https://doi.org/10.1109/ISADS.2013.6513420
12. Liao, H. J., Lin, C. H. R., Lin, Y. C., & Tung, K. Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications, 36*(1), 16–24. https://doi.org/10.1016/j.jnca.2012.09.004
13. Clincy, V., & Shahriar, H. (2018). Web application firewall: Network security models and configuration. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)* (pp. 835–836). IEEE. https://doi.org/10.1109/COMPSAC.2018.00144
14. OWASP Foundation. (2025). *OWASP Top 10:2025-Introduction*. https://owasp.org/Top10/2025/0x00_2025-Introduction/
15. Park, S., Kim, M., & Lee, S. (2018). Anomaly detection for HTTP using convolutional autoencoders. *IEEE Access, 6*, 70884-70901. https://doi.org/10.1109/ACCESS.2018.2881003
16. Davis, J., & Goadrich, M. (2006). The relationship between precision-recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine Learning (ICML '06)* (pp. 233-240). https://ftp.cs.wisc.edu/machine-learning/shavlik-group/davis.icml06.pdf
17. Buczak, A. L., & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials, 18*(2), 1153-1176. https://doi.org/10.1109/COMST.2015.2494502
18. Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE, 10*(3), e0118432. https://doi.org/10.1371/journal.pone.0118432
19. Lavin, A., & Ahmad, S. (2015). Evaluating real-time anomaly detection algorithms: The Numenta Anomaly Benchmark. In *2015 14th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 38-44). IEEE. https://doi.org/10.48550/arXiv.1510.03336
20. Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the International Conference on Information Systems Security and Privacy (ICISSP 2018)* (pp. 108–116). https://www.scitepress.org/papers/2018/66398/66398.pdf
21. National Institute of Standards and Technology. (2022). *Security and privacy controls for information systems and organizations* (NIST Special Publication 800-53, Rev. 4). https://doi.org/10.6028/NIST.SP.800-53r4
22. Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE Symposium on Security and Privacy* (pp. 305–316). IEEE. https://doi.org/10.1109/SP.2010.25
23. Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., & Ahmad, F. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies, 32*(1), e4150. https://doi.org/10.1002/ett.4150
24. Li, J., Zhang, H., & Wei, Z. (2020). The weighted Word2Vec paragraph vectors for anomaly detection over HTTP traffic. *IEEE Access, 8*, 141787-141798. https://doi.org/10.1109/ACCESS.2020.3013849
25. Vartouni, A. M., Kashi, S. S., & Teshnehlab, M. (2018). An anomaly detection method to detect web attacks using stacked auto-encoder. In *2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS)* (pp. 131-134). IEEE. https://doi.org/10.1109/CFIS.2018.8336654

26. Tolkachov, M. Yu., Dzheniuk, N. V., Zakharzhevskyi, A. H., Pohasii, S. S., & Hlukhov, S. I. (2024). Method of protecting information resources based on a semiotic model of cyberspace. *Suchasnyi Zakhyst Informatsii, 1*(57), 57-68. https://doi.org/10.31673/2409-7292.2024.010007
27. Tolkachov, M., Dzheniuk, N., Yevseiev, S., et al. (2024). Development of a method for protecting information resources in a corporate network by segmenting traffic. *Eastern-European Journal of Enterprise Technologies, 5*(9(131)), 63-78. https://doi.org/10.15587/1729-4061.2024.313158
28. OWASP Foundation. (2021). *OWASP Top 10:2021*. https://owasp.org/Top10/2021/
29. Cisco. (2020). *Cisco annual Internet report (2018-2023)*. https://www.deeprogram.org/library-v2/cisco-annual-internet-report-2018-2023-/5H7BUBW2
30. Sandvine. (2024). *The global Internet phenomena report 2024*. https://www.sandvine.com
31. International Organization for Standardization. (2023). *ISO/IEC 27032:2023. Information technology-Cybersecurity-Guidelines for Internet security*. ISO. https://cdn.standards.iteh.ai/sist-preview/76070/be57667fdd0b432490c253ca538c9938/ISO-IEC-27032-2023.pdf
32. Milevskyi, S. V., Kostiak, M. Yu., Milov, O. V., & Pohasii, S. S. (2019). Means of modeling agent behavior in information and communication systems. *Systems of Navigation, Management, Communication and Control, 6*(58), 63-70. https://doi.org/10.26906/SUNZ.2019.6.063

Отримано редакцією журналу / Received: 21.01.26
Прорецензовано / Revised: 05.02.26
Схвалено до друку / Accepted: 24.09.26