

Machine Learning for Cybersecurity: Techniques, Tools, and Real-World Applications

Yanping Zhang ^{1,✉}, Jianjun Yang ², Songqing Yue ³

1. Department of Computer Science, Gonzaga University, Spokane, USA

2. Department of Computer Science and Cybersecurity, University of North Georgia, Oakwood, USA

3. Department of Computer Science and Software Engineering, University of Wisconsin–Platteville, Platteville, USA

Received: April 30, 2025 | Review began: June 17, 2025 | Review ended: April 27, 2026 | Published: April 28, 2026

© Copyright 2026

This is an open access article distributed under the terms of the Creative Commons Attribution License CC-BY 4.0., which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Abstract

Cybersecurity has become a critical concern in the digital world, demanding robust and intelligent systems capable of defending against increasingly complex and evolving threats. This paper comprehensively explores the intersection between machine learning (ML) and cybersecurity, highlighting how data-driven approaches are reshaping threat detection, prevention, and response mechanisms. We provide an in-depth overview of core ML paradigms and analyze their relevance and effectiveness in various cybersecurity applications. A systematic literature review was conducted using major academic databases including IEEE Xplore, ACM Digital Library, SpringerLink, and ScienceDirect, covering peer-reviewed journal articles and conference papers published primarily between 2018 and 2025. Studies were selected based on relevance to ML-driven cybersecurity applications, methodological rigor, and demonstrated practical impact, while works lacking empirical validation or cybersecurity relevance were excluded. The review examines core ML paradigms: supervised, unsupervised, semi-supervised, and reinforcement learning, and analyzes their effectiveness across key cybersecurity domains such as malware detection, phishing detection, financial crime detection, anomaly detection, mobile security, and intrusion detection systems. We review prominent ML algorithms and detail their application in detecting malicious behavior across network traffic, system logs, and user activities. In addition, we survey open-source tools and platforms commonly used in cybersecurity-focused ML research and practice. The practical component of the paper showcases real-world use cases, including spam filtering, fraud detection, and endpoint security, illustrating how organizations can integrate ML techniques into operational security infrastructures. The paper also investigates the challenges and limitations of applying ML in cybersecurity, including adversarial ML, data imbalance, concept drift, and model interpretability. Finally, emerging research directions such as explainable AI, edge AI, federated learning, and quantum ML are discussed as promising avenues for advancing robust, transparent, and future-ready cybersecurity solutions. By bridging theoretical foundations with real-world practices, this paper aims to equip researchers, practitioners, and decision-makers with a holistic understanding of the capabilities, tools, and future potential of ML in advancing cybersecurity resilience.

Categories: Network Security, Computational Science and Engineering, Machine Learning (ML)

Keywords: cybersecurity, machine learning, cyber-threats, phishing, malware, mobile security, adversarial ml

Introduction And Background

In today's highly interconnected world, cybersecurity has become a crucial element for protecting individuals, organizations, and governments from cyberthreats. Cybersecurity refers to the protection of computer systems, networks, and data from digital attacks. It encompasses a wide range of technologies, practices, and processes designed

How to cite this article:

Zhang Y, Yang J, Yue S (April 28, 2026) Machine Learning for Cybersecurity: Techniques, Tools, and Real-World Applications. Cureus J Comput Sci 3 : es44389-025-00059-7. DOI <https://doi.org/10.7759/s44389-025-00059-7>

to safeguard the confidentiality, integrity, and availability of information. As security measures become more sophisticated, the detection of new cyberattacks becomes increasingly challenging. Over the past decade, detection strategies powered by machine learning (ML) and deep learning have become effective and popular. Cybersecurity and ML are increasingly intertwined. The integration of ML into cybersecurity presents a new era of defense mechanisms. ML has been used to enhance cybersecurity by automating threat detection, identifying anomalies, and predicting potential attacks based on historical data, which offers a wide spectrum of services and enables the system to not only reactively respond to threats but also proactively anticipate and mitigate them. Equipped with proactive measures and predictive capabilities, organizations can better ensure the integrity and security of invaluable data assets and fortify their digital infrastructure.

Various ML techniques have been used for cybersecurity, including clustering, classification, and regression trees. Phishing, malware, and spam are the most common types of cyberattacks, and ML techniques have been applied to detect them with high accuracy and low false-positive and false-negative rates [1]. As cyberattacks become more sophisticated, it is essential to continue developing and improving ML techniques to stay ahead of these threats. As subfields of artificial intelligence (AI), ML and deep learning are proficient in analyzing vast datasets and locating patterns, which could potentially change the way of dealing with cyberthreats fundamentally [2]. Various ML methods, like support vector machines (SVM) and decision trees (DT), can be used to identify various patterns of unexpected behavior [3]. While ML techniques can effectively detect well-defined features [4], deep learning strategies have the advantage of extracting features from huge datasets. Taking advantage of those strategies, we could potentially provide solutions for not only detecting potential threats but also anticipating new attacks, which enables us to be one step ahead of the rapidly evolving cyberthreats.

ML and cybersecurity

Due to the need to process an increasing amount of raw data within a limited timeframe, ML algorithms have become more and more ubiquitous. ML trains an algorithm to recognize patterns from datasets for use. They can automate and speed up the workflow. After understanding the pattern of the data, the algorithm can be used to predict outcomes with the input of new data. In the context of cybersecurity, ML can be used to recognize known attacks, detect vulnerabilities in networks, and identify potential attacks, which could help prevent attacks from being carried out. It could also be used to predict new attack patterns. ML consists of several approaches to training algorithms, with the most common for cybersecurity being supervised and unsupervised learning.

Supervised learning [5] uses labeled data to train an ML algorithm. The trained algorithm can recognize a certain behavior based on how it previously classified the training set. The benefit of supervised learning is that it teaches the algorithm how to predict or detect cyberattacks based on datasets of previous attacks. It excels at recognizing known attacks in networks as well as identifying vulnerabilities in the network of an organization. A challenge with supervised learning is the lack of readily labeled data.

Unsupervised learning [1,5] uses unlabeled data to find patterns with no known outcome. The ML algorithm will find structure, patterns, or relationships in the dataset. This approach is used for cybersecurity to detect new attack patterns or utilize anomaly detection as it creates a baseline of what the normal behavior of a system or program is and identifies abnormal behaviors.

Semi-supervised learning [5] combines elements of both supervised and unsupervised learning. Since labeled datasets are often scarce, this approach leverages the available labeled data to train the model while allowing it to identify patterns in unlabeled data. This can enhance the model's accuracy and broaden its potential applications. One common use of semi-supervised learning is in fraud detection. By integrating supervised and unsupervised techniques, this method strikes a balance between utilizing labeled data and reducing the cost of acquiring large labeled datasets.

Reinforcement learning [5] is a type of ML that relies on trial and error to learn from unlabeled data. It operates on a reward and penalty system, where the algorithm evaluates the outcomes of its actions and considers environmental factors to determine whether the action should be rewarded or penalized. This process closely mirrors human learning

How to cite this article:

and can enhance the reliability of ML models. Reinforcement learning helps the algorithm improve over time by rewarding it for correctly identifying patterns and penalizing it for incorrect ones, enabling it to handle new, similar data more effectively.

Advantages of ML in cybersecurity

ML models have their advantages for cyber detection. Without human intervention, ML algorithms can automatically detect patterns in large datasets, identifying potential security threats. This helps in early detection of cyberattacks like malware, phishing, or intrusion attempts. It reduces the burden on cybersecurity professionals, allowing them to focus on higher-level tasks and improving the efficiency of security operations. While datasets increase over time, ML models can keep improving the accuracy of detection through learning from past data. With the adaptability to predict new attacks and recognize previously unseen threats, ML models can greatly improve defense against sophisticated cyberattacks like zero-day exploits.

Challenges of ML in cybersecurity

Meanwhile, challenges for implementing ML-based cyber detection systems also bring concerns. Firstly, ML models require large amounts of high-quality, labeled data. It could be difficult to obtain this data in cybersecurity due to various factors, for example, privacy concerns and the ever-changing nature of threats. Secondly, processing sensitive data is always challenging, especially striking the balance between effective threat detection and respecting user privacy. Thirdly, it also requires significant computational resources to train complex ML models, which is expensive and time-consuming. This could delay the detection and response to threats in time. Fourthly, ML models could become too specialized in detecting certain types of attacks based on past data, which reduces their generalization ability and makes them less effective against new or unknown threats. Furthermore, ML models often employ deep learning algorithms, which operate as "black boxes," meaning their decision-making processes are not transparent. This lack of explainability makes it challenging to understand why a certain security decision was made and troubleshoot issues. Lastly, cybercriminals may take advantage of ML systems by training their own models to bypass security measures. Adversarial ML can deceive an ML model, causing it to incorrectly identify malicious activity as harmless.

Review

ML in cybersecurity

While cyberthreats have become a significant concern in today's digitally connected world, ML has been increasingly utilized in the field of cybersecurity to detect and prevent various cyberattacks, including misuse of resources, user access compromise, root access compromise, web access compromise, malware attacks, and denial-of-service attacks. ML algorithms are designed to analyze vast amounts of data to identify patterns and anomalies. This helps detect potential threats before they cause harm.

Moreover, ML can be used to continuously learn and adapt to new types of attacks, ensuring that cybersecurity measures remain effective. ML has played a significant role in improving cybersecurity. As cyberattacks become increasingly sophisticated, it is essential to continue developing and improving ML techniques to stay ahead of these threats.

In this section, we provide an in-depth review of recent top-tier, peer-reviewed journal articles and conference papers in the field of ML in cybersecurity. Our focus is on exploring the broad applications of ML within this domain. The following sections are organized according to the most prominent application areas.

ML for Malware Detection

Malware represents one of the most significant and evolving cybersecurity challenges today, affecting individuals, corporations, governments, etc. Malware, short for malicious software, refers to various types of harmful programs designed to disrupt, damage, or gain unauthorized access to computer systems. It includes viruses, worms, trojans, ransomware, spyware, and rootkits, each capable of compromising security and causing significant financial and operational damage. Traditional signature-based and rule-based malware detection methods, while effective against

How to cite this article:

Zhang Y, Yang J, Yue S (April 28, 2026) Machine Learning for Cybersecurity: Techniques, Tools, and Real-World Applications. Cureus J Comput Sci 3 : es44389-025-00059-7. DOI <https://doi.org/10.7759/s44389-025-00059-7>

known threats, struggle to identify new and evolving malware variants. As cyberthreats grow in sophistication, ML has emerged as a powerful tool for malware detection, enabling adaptive and intelligent threat detection beyond traditional methods.

As malware becomes more sophisticated, it has adapted to evade detection, particularly through the development of polymorphic malware. Polymorphic malware is especially dangerous because it continuously modifies its identifiable attributes, making traditional signature-based detection methods ineffective. Polymorphic malware can circumvent by altering its code structure or behavior without affecting its core functionality.

ML enhances malware detection by learning patterns from vast datasets containing both benign and malicious software behaviors. Unlike static detection techniques that rely on predefined malware signatures, ML-based approaches analyze multiple features of files, processes, and network behaviors to identify anomalies and previously unknown malware. These models continuously learn and improve, making them particularly effective against polymorphic and zero-day malware, which frequently modify their code to evade signature-based detection.

Various ML techniques are utilized in malware detection. Supervised learning models, such as DT, random forests (RF), SVM, and deep neural networks, classify files or processes as either malicious or benign based on labeled training data. These models use static and dynamic analysis features, such as opcode sequences, application programming interface (API) calls, and network traffic patterns, to make accurate predictions. In contrast, unsupervised learning approaches, such as clustering algorithms and autoencoders, are particularly useful for anomaly detection, identifying suspicious activities that deviate from normal behavior without requiring labeled datasets.

More recently, to address these limitations, ML techniques have become increasingly essential in malware detection. ML offers a dynamic approach, capable of detecting previously unseen threats by learning complex patterns in data [6]. The methods assess the effectiveness of various ML classifiers, including DT, convolutional neural networks (CNN), and SVM, which have shown promise in distinguishing between malicious and benign software. These methods enable the detection system to examine both static features (like code structure and metadata) and dynamic features (such as behavior in a virtual environment). By leveraging these multifaceted data points, ML models can enhance detection accuracy while reducing the incidence of false positives, thus offering a more robust approach against evolving malware. ML in cybersecurity has been focused on countering advanced threats like zero-day and polymorphic malware, which are notoriously challenging to detect.

Traditional detection techniques, which rely on predefined signatures, are often ineffective against these threats, as the malware's continually changing characteristics elude these static detection patterns. Automated system-level malware detection through ML presents a compelling alternative, as it adapts to changes and can even predict potential threats based on past behaviors. Various ML techniques, such as Bayesian algorithms [7] and DT [8], demonstrated effectiveness in detecting malware by utilizing diverse feature extraction and selection methods to implement more precise detection processes.

Another valuable approach in this context is dynamic detection techniques, which analyze the behavior of potentially malicious software in a controlled environment, such as a sandbox. By observing how malware behaves in real time, such as tracking API calls, memory usage, or network activity, these techniques can capture patterns indicative of malware. The dynamic data is then transformed into feature vectors for use in ML models. Classifiers like K-nearest neighbors (KNN) and RF have proven effective in this type of behavioral analysis, with some studies reporting detection accuracies approaching 100% in controlled experimental settings [9].

For even greater detection, advanced models that combine CNN and long short-term memory (LSTM) networks offer an ensemble approach that leverages both spatial and temporal features within the data. The CNN-LSTM model achieves high detection accuracy for IoT malware by combining CNN's ability to extract local spatial patterns with LSTM's capability to model temporal dependencies in API call sequences. However, CNN-based components are inherently limited to short-range feature learning and cannot effectively capture global execution semantics of complex, multi-stage malware. While LSTM alleviates this issue by learning long-term dependencies, it still suffers from sequential processing overhead and limited scalability for high-throughput, real-time systems. In contrast, transformer-based

How to cite this article:

Zhang Y, Yang J, Yue S (April 28, 2026) Machine Learning for Cybersecurity: Techniques, Tools, and Real-World Applications. Cureus J Comput Sci 3 : es44389-025-00059-7. DOI <https://doi.org/10.7759/s44389-025-00059-7>

models can model both short- and long-range dependencies simultaneously using self-attention and offer better parallelism, making them more suitable for large-scale and evolving malware detection. Moreover, the CNN-LSTM evaluation is conducted on a balanced dataset, which may not fully reflect real-world class imbalance and adversarial conditions, potentially overstating deployment performance [10].

To further enhance detection, a recent study proposed hybrid frameworks that utilize CNNs in combination with other ML algorithms to optimize the detection process. Such hybrid models capitalize on the strengths of different ML techniques, yielding improved detection rates and addressing the limitations inherent to single-method approaches. For instance, a study applied this hybrid method to Windows portable executable files, achieving high detection rates by drawing on multiple classifiers. By fusing these models, the framework not only improves detection accuracy but also mitigates the risks associated with relying on a single detection method [11].

ML has advanced malware through various sophisticated techniques that go beyond traditional signature-based methods. By harnessing static and dynamic data, feature selection, and hybrid modeling, ML enables robust, adaptable, and precise malware detection frameworks that continue to evolve alongside the threats they aim to mitigate.

ML has become a powerful tool in the field of malware detection, yet many open research challenges remain. One critical area is adversarial ML. In evasion attacks, malware authors manipulate features of malicious samples to bypass ML-based detection systems. Researchers can address this by developing robust models resistant to adversarial attacks, employing techniques such as adversarial training, ensemble methods, and explainability-driven defenses.

Another pressing challenge is Zero-Day and Novel Malware Detection. Traditional ML approaches often rely heavily on historical data, which makes them less effective against previously unseen (zero-day) threats. To overcome this, researchers can explore ML-based behavioral analysis techniques, including the use of API call sequences and memory forensics, rather than relying solely on static signatures.

A third promising direction is the application of graph theory and ML to design graph-based malware detection schemes. For example, graph neural networks can be used to model control flow graphs of binaries, system call dependency graphs, or the spread of malware across networks.

Overall, ML for malware detection is a rapidly evolving field characterized by an ongoing arms race between attackers and defenders. Future research will likely emphasize robustness, explainability, real-time detection, and adaptive learning in order to keep pace with increasingly sophisticated threats.

In order to provide a quick overview of this subsection, Table 7 lists the strengths and weaknesses of each technique of ML for malware detection.

Paper	Main Technique	Strengths	Weaknesses
Akhtar and Feng [6]	Classical ML (DT, SVM, RF, NB)	High accuracy; simple and efficient; good model comparison	Needs labeled data; weak against new malware; no real-time analysis
Vargafik et al. [8]	Resource-aware DT ensemble	Very low memory and computation; fast detection	Requires labeled anomalies; added system complexity; not malware-specific
Akhtar and Feng [9]	Sandbox-based dynamic ML	Robust to obfuscation; automatic features; very high accuracy	Sandbox is slow; unsuitable for real-time use; possible overfitting
Akhtar and Feng [10]	CNN-LSTM hybrid model	Automatic features; captures time patterns; high accuracy	High computation cost; low interpretability; unclear IoT deployment cost
Ahmad et al. [11]	Optimized SVM (PSO, ARO, NRO)	Improved SVM accuracy; good multi-dataset results	High training cost; limited scalability for large data

TABLE 1: Comparison of ML Techniques for Malware Detection

ML, Machine Learning; DT, Decision Tree; SVM, Support Vector Machine; RF, Random Forest; NB, Naive Bayes; CNN-LSTM, Convolutional Neural Network – Long Short-Term Memory; PSO, Particle Swarm Optimization; ARO, Artificial Rabbits Optimization; NRO, Nuclear Reaction Optimization; IoT, Internet of Things

ML for Phishing Detection

Phishing is a form of cyberattack where attackers deceive individuals into revealing sensitive information, such as usernames, passwords, and financial details, by impersonating legitimate entities. These attacks are often executed through fraudulent emails, fake websites, and malicious links. As phishing techniques become more sophisticated, traditional rule-based detection methods struggle to keep pace, leading to an increased reliance on ML for more effective and adaptive phishing detection.

ML offers a powerful solution for identifying phishing attacks by analyzing large datasets and detecting patterns that indicate malicious activity. Unlike conventional rule-based systems that rely on static blacklists and predefined heuristics, ML-based phishing detection systems can dynamically learn and adapt to evolving threats. This adaptability allows ML models to identify previously unseen phishing attacks by recognizing underlying behavioral patterns rather than just known attack signatures.

Several ML techniques are commonly employed in phishing detection, each suited to different detection domains such as email analysis, URL classification, and website content evaluation. Supervised learning approaches, which rely on labeled datasets, include models like LR, DT, SVM, and neural networks. These models classify emails, URLs, or web pages as either phishing or legitimate based on learned features. Among them, RF and deep learning-based neural networks have demonstrated high accuracy in identifying phishing attempts. Meanwhile, unsupervised learning techniques, such as clustering algorithms and autoencoders, are useful for anomaly detection, identifying potential phishing attacks without requiring labeled training data. Recent advancements in deep learning, including recurrent neural networks

(RNNs), LSTM models, and transformers like Bidirectional Encoder Representations from Transformers, have further improved phishing detection by enabling sophisticated natural language processing (NLP) and sequence analysis of phishing emails and URLs.

ML models rely on multiple extracted features to differentiate phishing from legitimate content. These features are categorized into URL-based features, which include characteristics like URL length, special character usage, and domain age; content-based features, such as misleading words like "urgent" or "verify" and embedded JavaScript within webpages; email header features, which analyze sender domain authenticity, recipient count, and authentication checks like SPF/DKIM; and network-based features, including IP reputation, domain registration details, and SSL certificate validity. By combining these features, ML models can effectively identify phishing attempts across different attack vectors.

Despite their effectiveness, ML-based phishing detection systems face several challenges. Adversarial attacks pose a significant threat, as attackers continuously evolve their phishing tactics to bypass ML models. Additionally, data imbalance remains a critical issue since phishing datasets typically contain far fewer phishing samples than legitimate ones, requiring advanced techniques like oversampling or anomaly detection to improve classification performance. Real-time detection also presents challenges, as deploying ML models efficiently in live environments demands significant computational resources.

To address these challenges, researchers are exploring hybrid models, reinforcement learning, and federated learning to enhance phishing detection accuracy and adaptability. By integrating these advancements, ML-based systems can become more resilient against emerging phishing threats, ensuring stronger cybersecurity defenses. As phishing attacks continue to evolve, leveraging ML will remain a crucial strategy in combating this persistent cyber threat.

Kapan and Sora Gunal [12] developed an improved phishing attack detection with ML, which focuses on enhancing phishing attack detection using ML technologies. Unlike traditional blacklisting methods that struggle with newly generated phishing websites, their ML-based solutions can adapt and improve detection accuracy. They introduced a new dataset of 1,000 websites (500 phishing and 500 legitimate) created from URLs collected from PhishTank and Alexa. Each website is represented by 25 features categorized into URL-based, HTML-based, and HTTP-based attributes.

Further, six well-known classifiers are evaluated: DT, KNN, SVM, Naive Bayes (NB), Multi-layer Perceptron, and Stochastic Gradient Descent. The performance of each classifier is assessed using five key metrics: accuracy, precision, recall, F1-score, and classification time. Feature sets are tested in different combinations to identify the best-performing configuration. Their research shows that the DT classifier, using URL + HTTP features, achieves the highest F1-score (0.99), accuracy (0.99), and precision (0.99) while maintaining one of the fastest classification times. In contrast, the NB classifier consistently performs the worst in terms of accuracy and F1-score, particularly when using only URL features. The study also compares its results to two external benchmark datasets, reaffirming the robustness of the DT classifier for phishing detection tasks. The authors conclude that selecting the right combination of features and classifiers is critical for effective phishing detection and suggest that future research should explore additional features and conduct cost analyses for real-time implementations. Their research indicates an open research on considering multiple-factors when developing phishing attack detection mechanisms.

Aslam et al. [13] addresses the growing threat of phishing attacks, which exploit the similarity between legitimate and malicious websites to deceive users. Traditional ML-based systems struggle with evolving phishing tactics due to reliance on predefined features. Their mechanism introduces AntiPhishStack, a novel two-phase stack generalization model aimed at overcoming these challenges.

Their approach has two phases. In Phase 1, the model extracts URL and character-level term frequency-inverse document frequency features from datasets obtained from PhishTank and Alexa. These features are processed through ML classifiers, such as the SVM, DT, and NB, using k-fold cross-validation to generate robust mean predictions. Phase 2 employs a two-layered LSTM network optimized with five adaptive optimizers: AdaDelta, Adam, RMSprop, AdaGrad, and

How to cite this article:

Zhang Y, Yang J, Yue S (April 28, 2026) Machine Learning for Cybersecurity: Techniques, Tools, and Real-World Applications. Cureus J Comput Sci 3 : es44389-025-00059-7. DOI <https://doi.org/10.7759/s44389-025-00059-7>

Stochastic Gradient Descent. These optimizers tune the model for dynamic prediction and ensure that it generalizes well to unseen data. The predictions from both phases are merged and refined using a meta-classifier (XGBoost) for the final robust prediction.

Their research shows that AntiPhishStack achieves an accuracy of 96.04%, outperforming traditional phishing detection models on multiple benchmark datasets. The proposed model operates independently of prior feature knowledge and does not rely on third-party services, such as domain age or page rank, which makes it more adaptable to real-world scenarios. AntiPhishStack's ability to detect zero-day phishing attacks using only URL-based features without manual intervention highlights its practical relevance.

Bezerra et al. [14] presented a case study on phishing detection with an ML net, which focuses on developing an efficient ML-based model to detect phishing emails in a highly imbalanced dataset. The case study was conducted in collaboration with E-goi, a Portuguese multi-channel marketing automation platform. Phishing detection is crucial for companies like E-goi because phishing campaigns can exploit the platform to distribute fraudulent emails, leading to serious reputational damage and service disruptions.

The dataset used for this research included over 500,000 email samples, of which only 0.1% were phishing emails. Their work discusses feature extraction, focusing on characteristics such as URL structure, presence of images, email content, and metadata like email sending time and font properties. Due to the significant class imbalance, the authors explored multiple under-sampling techniques, including clustering-based approaches, to balance the dataset and improve the model's performance.

Several ML models were tested in their work, including LR, SVM, DT, and ensemble models like RF and Gradient Boosting. The RF and neural network models showed the best results. The final model was a neural network with two hidden layers that achieved an F1-score of approximately 91% and detected over 87% of phishing emails while maintaining a low false positive rate of around 4.6%.

Feature importance analysis revealed that text-based features, email sending time, and email address length were key factors in identifying phishing emails. The study highlights that high-quality feature extraction and appropriate sampling techniques are critical for handling imbalanced data in phishing detection. Their work also suggested future improvements, such as detecting fake user profiles and developing anti-spam recommendation systems to enhance email security further.

Sumo et al. explored the application of ML techniques for detecting phishing websites, addressing the growing threat of cyber-attacks that exploit user trust [15]. The study evaluates four ML models, DT SVM, artificial neural network (ANN), and RF, using a dataset sourced from PhishTank.org.

Pre-processing techniques, such as artifact removal, normalization, and data encoding, were applied to enhance model accuracy. The DT model achieved the highest accuracy at 96.7%, followed by RF at 95.75%, ANN at 86.85%, and SVM at 83.8%. While DT and RF demonstrated strong classification capabilities, the ANN model showed potential with further fine-tuning. SVM underperformed due to challenges in handling phishing variations.

The study highlights the importance of integrating ML into cybersecurity frameworks for improved real-time phishing detection. It also acknowledges challenges such as overfitting in deep learning models and computational efficiency. Future research directions include hybrid model approaches, continuous learning techniques, and dataset diversification to adapt to evolving phishing tactics. The findings emphasize the need for adaptive cybersecurity solutions that leverage ML for enhanced phishing detection and mitigation.

ML for phishing detection is a dynamic and rapidly evolving research area, driven by the increasing sophistication of phishing attacks. Several key open research challenges remain in this domain. One important direction is the integration of multiple data sources to enhance detection accuracy. These data sources include: URL and Domain Features, such as lexical patterns, WHOIS data, and DNS records; Content Features, including HTML structure, JavaScript behavior, and embedded media; Email Headers and Metadata, which provide contextual information about the sender and routing paths.

How to cite this article:

Zhang Y, Yang J, Yue S (April 28, 2026) Machine Learning for Cybersecurity: Techniques, Tools, and Real-World Applications. *Cureus J Comput Sci* 3 : es44389-025-00059-7. DOI <https://doi.org/10.7759/s44389-025-00059-7>

A second major challenge is the development of real-time and low-latency detection mechanisms. Researchers are exploring the use of Edge AI to deploy lightweight ML models directly on browsers or mobile devices. Additionally, efforts are being made to design efficient feature extraction techniques that minimize computational overhead, enabling large-scale phishing detection systems, such as email filters, to operate effectively under resource constraints.

In order to provide a quick overview of this subsection, Table 2 lists the strengths and weaknesses of each technique of ML for phishing detection.

Paper	Technique	Strengths	Weaknesses
Kapan and Sora Gunal [12]	Feature + classifier study (DT, kNN, SVM, MLP)	Fast DT; public dataset; clear feature study	No new model; static features only; weak for zero-day
Aslam et al. [13]	Stacked ML + LSTM + XGBoost on URLs	High accuracy; no manual features; strong learning	Very complex; costly training; hard real-time use
Bezerra et al. [14]	Neural network for email phishing	Real company data; handles imbalance; good detection	Few phishing samples; slow training; manual features
Aldaham et al. [15]	DT, RF, SVM, ANN on PhishTank URLs	Simple models; good accuracy; easy deployment	Feature-dependent; weak for new attacks; limited real-time tests

TABLE 2: Comparison of ML Techniques for Phishing Detection

DT, Decision Tree; kNN, k-Nearest Neighbors; SVM, Support Vector Machine; MLP, Multilayer Perceptron; ML, Machine Learning; LSTM, Long Short-Term Memory; XGBoost, Extreme Gradient Boosting; URL, Uniform Resource Locator; RF, Random Forest; ANN, Artificial Neural Network

ML for Financial Crime Detection

ML has become a vital tool for detecting financial crimes, as traditional rule-based systems often fail to address the complexity and sophistication of modern cyberattacks. Financial crime detection involves identifying fraudulent activities, money laundering, insider trading, tax evasion, ransomware, and cryptocurrency-related offenses. The dynamic nature of financial crime requires adaptive and intelligent solutions, making ML well-suited for this domain.

Supervised learning models, such as DT, RF, and SVM, are widely used to classify suspicious transactions based on historical data. Unsupervised learning techniques, including anomaly detection and clustering, help uncover hidden patterns and unusual behavior without relying on labeled data. Deep learning methods, such as graph convolutional networks and RNNs, are particularly effective in detecting complex fraud scenarios, such as money laundering through intricate networks.

Group anomaly detection and hybrid models combining ML with graph analysis are gaining popularity to identify collaborative criminal networks and detect insider trading. Additionally, social network analysis is increasingly integrated to trace the relationships between suspicious entities. As financial cybercrime evolves, leveraging ML to enhance detection accuracy, reduce false positives, and improve model interpretability is essential for safeguarding financial systems from emerging threats.

Kumar et al. [16] addressed the challenge of predicting loan default and detecting financial crimes using ML algorithms. Due to the increasing demand for personal loans in developing countries and the rising number of bank fraud cases, there is a need for efficient methods to identify problematic candidates and minimize financial risks. The study utilizes a variety of supervised learning models, including DT, RF, KNN, SVM, Logistic Regression (LR), and NB. The primary goal is to minimize the false-positive rate to reduce non-performing assets and provide accurate predictions of customer behavior. The data for the study is obtained from the Lending Club dataset, containing financial records and customer details. Exploratory data analysis is performed to identify correlations and patterns in the data.

The authors evaluated the performance of different ML models based on accuracy, precision, recall, F1-score, and CAP curve analysis. Among the models, the SVM demonstrates the highest accuracy (83.40%) and zero false positives, making it the most effective model for fraud detection. The study highlights that, while KNN shows good accuracy, it has a high false positive rate. RF also exhibits high accuracy but has a moderate false positive rate.

The study emphasizes the importance of feature selection and data pre-processing to enhance model performance. It also suggests that combining behavioral and application features can improve the accuracy of fraud detection models. The authors conclude that incorporating ML-based models into banking systems can significantly enhance fraud detection and risk assessment, helping financial institutions make informed decisions.

The study points out that future research directions include leveraging social media data and big data analytics to enhance prediction accuracy. Additionally, integrating forensic accounting techniques and conducting further studies on financial fraud can offer better insights into detecting and mitigating fraudulent activities.

Darvishi et al. [17] proposed an ML-based framework to detect, isolate, and accommodate sensor faults in digital twins. The increasing reliance on sensors in Industry 4.0 makes accurate fault detection essential, as faulty sensors can lead to significant performance degradation and safety risks. The proposed architecture, called SFDIA, comprises three stages: virtual sensors, residual computation, and classification. Virtual sensors predict measurements from unreliable sensors by leveraging temporal correlations and measurements from other reliable sensors. Residual computation measures the difference between actual and predicted sensor values, generating residual signals that reflect inconsistencies. The classifier then uses these residual signals to detect and isolate faulty sensors. In cases of fault detection, the system replaces the faulty sensor's data with the predicted values from the virtual sensors, thus maintaining system reliability.

The study evaluated the SFDIA framework on three real-world datasets: air quality, wireless sensor networks, and permanent magnet synchronous motor data. The results demonstrate that the proposed architecture achieves high detection accuracy (up to 99.9%) and low false alarm rates, especially for bias faults. Furthermore, the system shows robustness in handling weak and strong fault signals, outperforming state-of-the-art methods like SVM and fully connected cascade neural networks.

The study concludes that the SFDIA architecture is highly effective for detecting weak and strong faults while maintaining low computational complexity. Future work may explore enhancing model performance using more complex architectures, such as CNN or RNN, to better capture temporal correlations in sensor data.

Cryptocurrencies have gained significant attention in financial markets, but their pseudonymous nature poses challenges for Anti-Money Laundering and Countering the Financing of Terrorism efforts. Pocher et al. [18] investigated ML-based forensic techniques to detect anomalous cryptocurrency transactions, aiming to strengthen financial security and regulatory compliance.

They explored various ML approaches, including supervised, unsupervised, and hybrid models, to identify suspicious transactions. Unsupervised learning methods, such as clustering and anomaly detection, prove effective in recognizing unusual patterns without prior labeled data. Meanwhile, supervised models leverage labeled datasets to classify transactions as legitimate or fraudulent, improving detection accuracy. They also evaluated hybrid approaches that combine both methods for enhanced predictive performance.

How to cite this article:

A key focus of the research is feature engineering, where transaction characteristics such as volume, frequency, and network behavior are analyzed to develop effective detection models. Their research highlights the importance of explainability in ML-based forensic investigations, ensuring that regulatory bodies can interpret model decisions. Additionally, their study acknowledges challenges such as data imbalance, adversarial attacks, and evolving fraud tactics, emphasizing the need for continuous model adaptation.

Empirical results indicate that ML-based methods significantly improve the detection of illicit transactions compared to rule-based systems. Their findings suggest that integrating AI-driven analytics into cryptocurrency monitoring can bolster Anti-Money Laundering and Countering the Financing of Terrorism frameworks, reducing financial crime risks.

Innan et al. [19] presented a comparative study of four Quantum Machine Learning (QML) models: Quantum Support Vector Classifier (QSVC), Variational Quantum Classifier (VQC), Estimator Quantum Neural Network (EQNN), and Sampler Quantum Neural Network (SQNN) for detecting financial fraud. The study leverages the BankSim dataset, which simulates bank transactions, including 7,200 fraudulent records out of 594,643 total transactions. Key features analyzed include transaction amount, customer demographics, and merchant categories, with "amount" identified as the most influential feature for fraud detection.

The QSVC model outperformed the others, achieving an F1-score of 0.98 for both fraud and non-fraud classes when using the ZFeatureMap for data encoding. This highlights the effectiveness of quantum kernels in measuring state similarities without traditional loss functions. The VQC also performed well, with an F1-score of 0.90, though it required optimization during training. In contrast, EQNN and SQNN showed lower accuracy, attributed to limitations in their quantum circuit designs and suitability for continuous data.

Their study underscores the potential of QML in fraud detection, emphasizing advantages like real-time data analysis, reduced computational complexity, and scalability through quantum parallelism. However, challenges such as hardware limitations and the need for more efficient algorithms remain. The findings suggest that QML, particularly QSVC, could revolutionize fraud detection by offering superior performance over classical methods, paving the way for future advancements in quantum computing applications for financial security.

ML for financial crime detection is a vital and growing area of research, driven by the increasing complexity of fraud, money laundering, and other illicit activities. Several key research directions have emerged in this field.

One important area is the development of AI-based fraud prevention mechanisms, including the deployment of lightweight models on mobile devices and ATMs to enable real-time, resource-efficient detection.

Another critical direction is the design of AI-assisted anti-money laundering systems. This includes advanced risk scoring techniques, such as dynamic customer risk profiling based on transaction history, behavioral patterns, and social network analysis. Additionally, ML can help reduce false positives in anti-money laundering alerts, improving efficiency and decision-making in compliance operations.

In order to provide a quick overview of this subsection, Table 3 lists the strengths and weaknesses of each technique of ML for phishing detection.

Paper	Technique	Strengths	Weaknesses
Kumar et al. [16]	DT, RF, KNN for loan fraud and default prediction	Simple ML models; low false positives; easy to deploy	Needs labeled data; weak for new fraud patterns
Darvishi et al. [17]	Neural-network virtual sensors for fault detection and isolation	Handles weak sensor faults; works for digital twins	Complex structure; high training cost
Pocher et al. [18]	Graph Neural Networks (GCN, GAT) on Bitcoin transaction graphs	Strong for crypto AML; captures transaction relations well	Graph building is costly; limited to blockchain data
Innan et al. [19]	Quantum ML (QSVC, VQC, QNN) for credit fraud	Very high accuracy; explores new QML direction	Needs real quantum hardware; high cost; not production-ready

TABLE 3: Comparison of ML Techniques for Financial Crime Detection

DT, Decision Tree; RF, Random Forest; KNN, k-Nearest Neighbors; GCN, Graph Convolutional Network; GAT, Graph Attention Network; QSVC, Quantum Support Vector Classifier; VQC, Variational Quantum Classifier; QNN, Quantum Neural Network; AML, Adversarial Machine Learning; QML, Quantum Machine Learning

ML for Anomaly Detection

Anomaly detection is a long-standing and critical technique in cybersecurity, used to uncover deviations from normal behavior that could indicate security threats such as intrusions, malicious activities, or policy violations. Unlike signature-based detection (which relies on known attack patterns), anomaly-based detection builds a model of normal operations and flags outliers without prior knowledge of the attack signature [20]. ML has become a cornerstone in this area, as it excels at modeling complex patterns and identifying subtle abnormalities in large datasets. In recent years, the field has evolved from relying on traditional ML algorithms - such as clustering and one-class SVMs - to adopting more advanced techniques, including deep learning, ensemble methods, and graph-based approaches for anomaly detection.

In cybersecurity, anomalies often correspond to potential attacks or breaches that do not match the expected behavior of the system or the user. Examples include unusual network traffic flows which could signal a network intrusion or scanning activity, deviations from user login patterns (possibly indicating account compromise or insider misuse), or irregular system calls by a process (a sign of malware). Anomaly detection systems serve as the backbone of many intrusion detection systems (IDS) and security information and event management tools, providing a second line of defense beyond preventive measures. The appeal of ML-based anomaly detection is its ability to detect novel or unknown attacks - a crucial advantage given the emergence of new threat variants that lack signatures.

Traditionally, anomaly detection in security involved statistical thresholds or expert-defined rules. ML techniques brought greater adaptability: models can be trained on historical benign data to learn complex normal behavior profiles. Common approaches include clustering algorithms (to group similar behaviors and detect outliers), classification models trained to distinguish normal versus malicious (if some labeled data are available) and one-class learning methods (like one-class SVM or autoencoders) that explicitly learn the boundary of normal data and identify anything outside that boundary as anomalous. A classic deployment is network anomaly detection: capture features of network traffic over

time and raise alerts when patterns deviate significantly from the learned baseline. Such anomaly-based network intrusion detection systems (NIDS) can potentially catch zero-day exploits and stealthy attacks that do not match any known malware signatures.

Research in the last decade underscored the importance of anomaly detection for critical infrastructure protection. Ferrag et al. [21] note that IDS are “part of the second defense line” in protecting critical networks, and that data mining and ML techniques can yield IDS with higher accuracy and robustness than traditional signature-based systems. Unlike misuse detection, which only finds known patterns, anomaly detection can identify “zero-day” incidents - albeit at the cost of higher false positive rates. This balance between sensitivity and false alarms is a recurring theme: because anomalies are rare by definition, ML models must be carefully tuned to avoid flooding analysts with false alerts. As we shall see, recent works strive to improve the trustworthiness of anomaly detectors, ensuring that flagged events truly merit attention.

Recent work has increasingly shifted anomaly detection toward representation learning directly on raw security telemetry, reducing reliance on hand-crafted statistics and improving generalization to evolving traffic. For network intrusion detection, transformer-based models have been proposed to capture temporal correlations and feature interactions in traffic streams and flow sequences, including architectures evaluated in cloud-security settings [22]. In parallel, self-supervised contrastive learning has become a common strategy for reducing dependence on labeled attack data; for example, transformer encoders pretrained with traffic augmentations can learn transferable representations that improve cross-dataset anomaly detection and downstream supervised intrusion detection [23]. At a larger scale, “foundation model” style pretraining on massive unlabeled traffic is emerging, where models are pretrained on packet and flow data and then fine-tuned for downstream security tasks under limited labels [24,25].

Graph-based anomaly detection is also gaining traction because many security signals are naturally relational, such as communication graphs between endpoints, process-resource graphs, or authentication graphs. Recent studies propose graph neural network pipelines that construct flow graphs and learn node or edge embeddings to detect anomalous communications [26,27]. Survey work has further systematized this area, highlighting key design choices in graph construction, message passing, and evaluation methodology for intrusion detection systems based on graph neural networks [28].

Another important current theme is anomaly detection under pervasive encryption, where payload inspection is unavailable and models must rely on flow-level timing and size statistics. Scientific Reports recently introduced ET-SSL, a self-supervised contrastive framework for detecting anomalies in encrypted traffic without decryption, illustrating the broader shift toward privacy-preserving anomaly detection [29]. Complementary work explores interpretability for encrypted-traffic anomaly detectors using post-hoc explanation methods such as SHapley Additive exPlanations, which can help analysts understand which traffic features drive alerts [30].

Beyond networks, recent anomaly detection research targets system logs and multivariate telemetry that feed modern security monitoring platforms. LogBERT is one representative approach that uses self-supervised transformer objectives to model normal log sequences and flag deviations [31]. For time-series telemetry (e.g., service metrics, industrial control sensors, or host counters), transformer-based detectors such as anomaly transformer and TranAD capture long-range temporal dependencies and have demonstrated strong performance on benchmark multivariate anomaly-detection datasets [32,33].

ML for Mobile Security

The wide popularity of smartphones and mobile apps has made mobile platforms a prime target for attackers. We survey how ML is used to protect mobile devices, focusing on Android malware detection using ML-enhanced static and dynamic analysis, mobile phishing detection, and other defenses. We highlight state-of-the-art detection techniques, from classical classifiers to deep neural networks, that achieve high accuracy against mobile threats. We also examine real-world usage of these techniques in commercial mobile security solutions and remaining issues like evolving malware tactics and privacy constraints.

How to cite this article:

Mobile security threats have exploded in number and variety in the last decade. Android, being the most widely used mobile operating system in the world, has seen millions of malware samples ranging from trojanized apps and spyware to ransomware and adware. Traditional defenses (e.g., signature-based scanners) struggle to keep up with the volume of new mobile malware variants, which can reach hundreds of thousands per year. Meanwhile, users frequently install apps from app stores or third-party sources without a thorough vetting. According to a McAfee report, 2020 was “the year of mobile sneak attacks”, with attackers finding new ways to hide malware and steal data from phones [34]. In this context, ML offers a way to automate and scale mobile threat detection: instead of relying on manually crafted signatures, ML models can learn patterns that distinguish malicious apps or behavior.

ML has proven particularly effective in anomaly-based detection on mobile devices. As Kouliaridis and Kambourakis [34] note, ML can model the normal behavior of mobile apps or systems and detect malware such as deviation. For example, a learning-based system might observe that a legitimate weather app should not send premium SMS messages; if it does, that is an anomaly likely caused by malware piggybacking on the app. In fact, many mobile malware detection solutions are essentially specialized anomaly detectors: they look for apps that behave differently than benign apps seen during training. Another major advantage is the variety of features that ML can handle. Android apps can be analyzed through multiple lenses, yielding different feature sets: a) static features extracted from the app’s code or manifest (e.g., permissions requested, API calls used, presence of suspicious strings, and opcode sequences), and b) dynamic features from runtime behavior (e.g., system calls made, network traffic generated, and battery usage patterns). ML models can be trained on any or all of these features to make a determination. Traditionally, static analysis is faster and does not require running the app (important for app store scanning), while dynamic analysis can catch runtime-only malicious behaviors (important for detecting logic bombs or evasive malware). ML allows combining these through hybrid analysis.

Static analysis features, such as permissions and code structures, have been effective against known malware but can be obfuscated, whereas dynamic analysis is more robust to obfuscation but is heavier to deploy [34]. Static features work well on older malware but fail when apps use encryption or packing to hide malicious code. Feeding dynamic behavior data that are obtained by running apps in a sandbox into ML models significantly can improve detection of new and obfuscated malware. However, dynamic analysis on mobile requires instrumentation or an emulator, which sophisticated malware can detect and evade and some malware stays dormant if it senses it is in an emulated environment [34].

The role of ML in mobile security is to provide adaptive, data-driven detection mechanisms that can keep pace with evolving threats. By learning from a large number of apps and device behaviors, ML models can generalize and catch malware that was not explicitly seen before. This has been validated in numerous studies, where ML-based detectors achieve high detection rates on contemporary malware datasets. Indeed, a survey by Odusami et al. [35] concludes that ML-driven approaches are more promising and achieves higher accuracy in detecting unknown Android malware compared to signature techniques.

Static analysis entails examining an Android app’s contents without executing it. This includes analyzing the app’s APK file, which contains Dalvik bytecode, manifest file, resource files, etc. Over the years, a rich set of features from these sources have been used to train ML classifiers including permissions and intents, API calls and code patterns, op-code and binary features, and app metadata and other attributes.

Android’s manifest declares what permissions an app requests, e.g., SEND_SMS, READ_CONTACTS, and what intents it listens for. Certain combinations of permissions can be strong indicators of malware, e.g., a simple wallpaper app requesting SMS sending and device admin permissions is suspicious. Early ML models like DREBIN [36] used permissions (among other features) in an SVM classifier to detect malware with high success. DT and RF models have likewise been trained on permission sets, yielding interpretable rules, e.g., “if app requests SEND_SMS and READ_CONTACTS and is not from the Play Store, classify as malware”.

By analyzing an app’s code, one can extract which Android API functions are invoked. Malware often calls certain APIs like cryptographic functions, reflection, dynamic code loading, or specific suspicious APIs such as sending SMS, recording audio, etc. Researchers create features such as “frequency of API X” or boolean “uses API Y”. More advanced approaches

How to cite this article:

represent the app's program as a call graph or sequence of API calls and then apply graph mining or sequence classification. ML models from SVMs to deep neural networks have been used where a notable approach converts the sequence of API calls into a vector (e.g., using word embedding techniques from NLP) and then feeds that to a classifier.

Treating the app binary as a sequence of machine instructions or opcode is another static method. N-gram analysis of opcodes can be done. Essentially, the app is represented by the distribution of opcode sequences of length n . This can then be input into algorithms like NB, SVM, or even CNNs by treating the opcode sequence like text. For example, a CNN can learn patterns of opcode usage that correlate with malicious behaviors. Some deep learning models actually treat the raw bytecode as input, similar to how one might detect malware in Windows executables by raw bytes. The approaches might be effective but tend to be less interpretable.

Features like the app's size, the presence of native (C/C++) code, the use of particular permissions combinations, strings found, e.g., URLs or suspicious substrings, and whether the app is packed or obfuscated can all be inputs to ML. Modern datasets like AndroZoo (<https://androzoo.uni.lu/>) [37] provide thousands of features per app. Some studies use more than 100 features, combining all the above categories.

ML classifiers commonly used on these static features include RF, SVM, KNN, NB, and increasingly deep learning models, where deep neural networks can learn feature representations automatically.

One key trend is the move toward automated feature learning via deep learning. Traditional approaches needed experts to decide which features matter. In contrast, deep learning, especially deep belief networks or autoencoders and now more so CNNs and RNNs, can ingest raw or high-dimensional features and figure out which aspects are discriminatory. Gibert et al. [38] review how classical ML was used for malware detection and then how deep learning became promising, explaining that deep learning can capture complex patterns that older methods missed. For example, a deep neural network might learn a subtle combination of API usage patterns that indicate malware, a combination too complex to encode as a simple rule. Many recent Android malware detectors indeed use deep fully connected networks on a feature vector of an app (where the vector might be thousands of binary indicators for various API calls or permissions) - these have achieved detection accuracies well above 90% on large malware datasets [39].

In summary, static analysis ML detectors are valuable because they can be deployed in an app store setting to scan large numbers of apps quickly without the need to run each app extensively. They have detected countless malware and potentially unwanted apps that would be hard to enumerate with signatures. However, they face challenges from malware authors: code obfuscation, encryption of payload, use of dynamic code loading or reflection, which makes static analysis incomplete. Hence, static detectors are increasingly complemented by dynamic analysis for a fuller picture, as we discuss next.

Dynamic analysis monitors an app's behavior at runtime to detect malicious actions. It typically involves running the app in a sandbox or emulator and collecting data such as system calls, API invocations, network traffic, memory usage, etc. ML models can then be trained on this behavioral data to recognize malware. Key types of dynamic features and ML approaches include system call sequences, API call traces and logs, network traffic analysis, and device and sensor behavior.

As an app runs, it requests services from the OS via system calls, e.g., open a file, send network data. Malware may invoke certain sequences of system calls not commonly seen in benign apps, for example, rapidly sending many network connections, or reading system files it normally would not. By tracing system calls, one can derive sequences or frequency counts, and apply sequence classification or anomaly detection. Hidden Markov Models and LSTM networks have been used to model normal against malicious system call patterns. A malicious app might, for instance, have a system call trace that an LSTM flags as deviating significantly from any benign sequence seen during training.

Similar to system calls but at the higher Android API level, one can log the sequence of important API calls during execution. This might include UI events, file accesses via API, cryptographic API usage, etc. Research by Wang and Yiu [40] has used RNNs to API call sequences to classify malware against benign. The intuition is similar to NLP via treating the

How to cite this article:

sequence of calls as a “sentence” and using an RNN/CNN to classify it. Such models have achieved high accuracy, especially when focusing on discriminative API calls, e.g., calls to send SMS, record audio, etc., which benign apps may rarely do in certain contexts.

Many mobile malware communicate with command-and-control servers or exfiltrate data. Monitoring an app’s network traffic during runtime can reveal anomalies, like connections to known bad domains or an unusual amount of data upload for a given app type. ML techniques from network anomaly detection apply here too. For example, a classifier could be trained to recognize traffic patterns of common benign app categories and flag an app whose traffic differs strongly (e.g., a flashlight app making frequent HTTP POST requests is suspicious). Shabtai et al. [41] in early work detected anomalies in app network traffic using clustering. More recent works use flow-based features and apply RF or SVM to classify traffic as malware-related or not.

Some malware especially spyware or stalkerware misuses device sensors (camera, microphone, GPS). Dynamic monitoring can catch apps that, say, turn on the microphone without user interaction or frequently fetch location in the background. ML models might incorporate these behaviors too. For example, a benign weather app might check location a few times a day. If an app claiming to be weather is pinging GPS continuously and sending data out, a dynamic anomaly detector can flag it.

Dynamic analysis often produces a rich set of time-series data. One approach is to derive summary statistics from these traces, e.g., number of file writes, number of distinct IPs contacted, etc., in a given time window and use those as features for a classifier. Another approach is to feed the raw or lightly processed traces into sequential models such as LSTMs or temporal convolutional networks to learn the pattern of behavior.

While dynamic analysis improves coverage of obfuscated malware, it introduces complexity in data collection. According to Kouliaridis and Kambourakis [34], dynamic systems are inherently more complex because they need infrastructure to execute apps and generate events, and they have to deal with malware trying to evade in sandboxes. However, many cutting-edge solutions use some form of dynamic monitoring. For example, the Android Malware Genome Project and other datasets have dynamic traces for malware which researchers have used to train deep learning models that outperform purely static ones.

Adversarial ML

As ML becomes ingrained in security defenses, attackers have not been left idle. An increasing number of real-world incidents show that adversaries can attack the ML models themselves. As an emerging discipline, adversarial ML examines how malicious actors can exploit vulnerabilities in ML algorithms and how defenders can harden models against such exploitation. Adversarial ML commonly shows itself in two forms: evasion attacks, where an attacker crafts inputs (e.g., malware samples, network traffic, images) designed to be misclassified by an ML model at test time; and poisoning attacks, where an attacker manipulates the training data or learning process to corrupt the model (e.g., inserting malicious samples during training so the model learns incorrect behavior). Additionally, model extraction and inference attacks target confidentiality by stealing or revealing sensitive aspects of models or training data. This section reviews adversarial ML with an emphasis on its implications for cybersecurity. We begin by outlining the threat models and attack techniques, and then review notable research findings on attacks against ML-based security systems such as evading malware classifiers or IDS. Our focus is on integrity attacks that affect detection and classification tasks.

In adversarial ML research, the common goals include 1) evasion that works by causing the model to misclassify a malicious input as benign, or false negative, 2) false alarm that results in a benign input being misclassified as malicious, or false positive, potentially affecting operations, and 3) model degradation that generally reduces the accuracy or confidence of a model. The main goal of the attacker is usually evasion, where malware authors want their malware to go undetected by an ML-based detector [42]. A secondary goal might be to induce false alarms to erode trust in the system or waste analysts’ time.

How to cite this article:

Evasion attacks occur at model inference time. The attacker takes an input that would normally be flagged and carefully modifies it so that the ML model outputs a benign classification. Importantly, the modifications are designed to be subtle enough that, for a human or for the normal operation of a system, the input still appears essentially malicious or has the same effect. In the image domain, such as attacking a classifier that identifies images, this often means adding small pixel perturbations, for example the famous “adversarial examples” where adding quasi-imperceptible noise to an image causes a deep network to misidentify it.

The consequences of successful evasion are serious. It means that malware, intrusions, or malicious content can slip past AI-driven defenses, potentially rendering them ineffective. A real example was the case of attackers who reportedly bypassed a popular ML-based antivirus by adding a large chunk of benign data to each malware file by exploiting the ML model's tendency to weight that benign section heavily in its decision, thereby misclassifying the file as safe. This was essentially an adversarial attack executed in the wild, forcing the vendor to retrain their model.

Evasion attacks may take various forms, such as malware evasion, network intrusion evasion, and spam and phishing evasion. Consider an Android malware classification model that uses API call frequencies as features. An attacker could add additional redundant API calls that are common in benign applications by effectively adding noise to the feature vector until the malware profile looks closer to benign in the eyes of the model [43]. For instance, if the model heavily weighs usage of certain suspicious APIs, the attacker might avoid those or pad usage of innocuous APIs to dilute the suspicious signals. Anderson et al. [43] demonstrated an automated approach using reinforcement learning to learn how to modify Portable Executable malware files to avoid a static malware detector. Their system learned a sequence of modifications, such as adding benign sections to the binary, manipulating header values, etc., which preserved the malware functionality but drastically reduced the detection rate of a leading ML antivirus engine. This highlights that even without knowing the exact model, one can train a substitute or use trial-and-error to find effective evasion strategy [43].

For anomaly-based NIDS, an attacker can craft network traffic that blends in. Rigaki and Garcia [44] showed that malware command-and-control traffic could be modified to mimic normal traffic (e.g., making botnet communication resemble benign HTTPS connections to Google) by using techniques, such as generative adversarial networks to automate this merging. Essentially, malicious traffic is modulated to follow patterns of legitimate traffic, causing the anomaly detector to view it as normal. This is akin to adding perturbations in feature space (timings, packet sizes, etc.) to shift the malicious traffic closer to the normal cluster.

Spammers have long altered their messages to avoid keyword-based filters. Modern spam filters use ML like Bayesian or neural classifiers in email text, so spammers now generate emails that appear to have harmless content, sometimes including chunks of legitimate text or random sentences to confuse the classifier [45]. These are adversarial examples in text. Phishing websites may incorporate content from the legitimate sites they are spoofing to confuse content-based detectors, or they might serve different content to the user and the security scanner, a form of adversarial evasion known as cloaking.

A key aspect of evasion is that it often leverages the model gradient in white-box cases to find which direction to change the input to make the model output what the attacker wants. The work of Szegedy et al. [46] on adversarial image examples first revealed how calculating the loss gradient with respect to input can guide minimal perturbations to cause misclassification. In [42], the authors had earlier applied gradient-based attacks to a linear spam classifier, altering feature values of a spam message along the gradient of least detection to evade it. We see an analogous use of gradients if the model is differentiable. For example, a gradient-based approach could highlight which bytes of a malware binary most influence the model. An attacker could then focus on altering those bytes, perhaps in a way that does not break the program. However, binary executables and discrete features pose challenges to direct gradient methods, so heuristic or reinforcement approaches are more common.

Even black-box attacks can be powerful. Attackers can train their surrogate model to imitate the target (using queries to the target to get labels), and then generate adversarial inputs on the surrogate, which often transfer to the target model. This has been demonstrated in malware detection where an attacker queries an online malware scanner with a variety of

How to cite this article:

modified malware samples to observe which get through, uses that data to train a local model approximating the scanner's behavior, and then crafts evading samples on that surrogate to use against the real scanner.

Poison attacks target the training process of the model. If an attacker can influence the training data, even partially, they may introduce specially crafted instances that skew the model's decision boundary. This could occur if the learning system consumes data from untrusted sources. For example, consider an IDS that retrains itself on what it believes are benign network logs. An attacker who intermittently generates malicious traffic that is not detected might have that malicious traffic included in the training set labeled "benign", gradually teaching the model that such traffic is normal, making it even less likely to detect it in the future. This is a form of feedback loop poisoning.

Another poisoning scenario is an attacker contributing to crowd-sourced threat intelligence. If a company uses community-shared malware feeds to train an ML model to recognize malware, an attacker could submit mislabeled samples, malware labeled benign or vice versa, to subvert the model. One particularly insidious form of poisoning is the backdoor attack. Here the attacking goal is not to degrade overall model performance, but to embed a hidden trigger in the model. They poison the training data with a small number of samples labeled in a certain way with a trigger pattern. For example, for an image classifier, they might put a specific pixel pattern on some images and label them as a certain class. In a malware context, a trigger could be a specific sequence of bytes or network behavior. Li et al. [47] survey backdoor attacks and note that an attacked model behaves normally on clean inputs, but when an input contains the secret trigger (a pattern chosen by the attacker), the model produces a specific attacker-desired output. In security terms, an attacker could hypothetically backdoor a malware detector such that any file with a specific byte sequence (the trigger) is always classified as benign. This would allow the attacker to then distribute malware embedded in that sequence to evade detection universally on systems using the compromised model.

Both poisoning and evasion underscore the need to treat the ML model as part of the "attack surface" in a system. Classic security assumes that models are static and trustworthy, but adversarial ML recognizes that attackers can target the model directly.

From a defensive perspective, mitigating adversarial ML requires a defense-in-depth approach across the ML lifecycle in data collection and labeling, training and evaluation, deployment, and ongoing operations rather than a single "silver bullet" technique [48]. The following practices are concrete measures that are commonly recommended in the adversarial ML and secure AI engineering literature:

(1) Robust training and robustness evaluation (evasion resistance): Robustness begins by explicitly defining the attacker's feasible perturbations (e.g., "functionality-preserving" PE modifications, API-call padding constraints, or traffic-shaping limits) and then training and testing against that threat model [48]. In settings where differentiable perturbations are meaningful, adversarial training based on robust optimization (e.g., PGD-style training) is a standard baseline that measurably improves worst-case robustness [49]. Methods such as TRADES formalize the trade-off between clean accuracy and robustness, which can be useful for tuning operational tolerances in security systems [50]. Importantly, defenses should be evaluated using strong, adaptive attacks to avoid "false security" from weak evaluations or gradient masking; ensemble evaluations such as AutoAttack are widely used for this purpose [51]. Where feasible, certified defenses (e.g., randomized smoothing) can provide formal robustness guarantees under specific perturbation norms, which is useful for high-assurance components [52].

(2) Calibration and safe decision policies (reducing overconfident failures): Since adversarial and out-of-distribution inputs can induce confident but wrong predictions, calibration should be treated as an operational safety control rather than a purely ML metric. Post-hoc temperature scaling is a lightweight, widely used practice to improve probability calibration without retraining [53]. For higher-stakes deployments, uncertainty-aware approaches such as deep ensembles can improve calibrated uncertainty estimates and provide a more reliable confidence signal for abstention or escalation [54]. Practically, defenders can (i) monitor calibration error (e.g., expected calibration error) on a clean validation stream, (ii) implement an "abstain/send-to-analyst" policy below a confidence threshold, and (iii) treat low-confidence or distributionally novel inputs as requiring secondary analysis (e.g., sandbox detonation for malware, additional sensors for NIDS) rather than forcing a hard decision.

How to cite this article:

(3) Monitoring, drift detection, and incident response (operational controls): Because attack strategies and benign baselines shift over time, ML-based defenses benefit from continuous monitoring similar to other security telemetry. Log model inputs and features (with appropriate privacy controls), predictions, confidence or uncertainty scores, and downstream outcomes; generate alerts for distribution shifts in key features; and track changes in false-positive and false-negative rates over time [48]. Out-of-distribution anomaly detection methods can be integrated as a “front-end” gate or as an additional signal; for example, Outlier Exposure improves detection of anomalous inputs by training in auxiliary outlier data [55]. Finally, robustness should be treated as a recurring security test activity: periodically re-run red-team style adversarial evaluations and regression-test robustness whenever the model, features, or data pipeline changes.

(4) Hardening the model via reducing extraction risk and narrowing the attack surface: Beyond integrity attacks, deployed ML security systems often expose APIs or services that can be queried by adversaries. Practical hardening measures include authentication, rate limiting, anomaly detection on query patterns, and minimizing information returned (e.g., avoid returning full confidence vectors unless needed) [56,57]. These controls are not complete defenses on their own. Model extraction can remain possible even when confidence values are removed - so they should be combined with monitoring and governance controls [56]. Where confidentiality of training data is a concern, defenses against inference attacks should also be considered. Membership inference attacks demonstrate that models may leak whether specific records were in training data, motivating privacy-aware training and controlled access to model outputs [58].

(5) Data pipeline security and defenses against poisoning and backdoor attacks: Because poisoning and backdoors exploit the training pipeline, defenders should secure data ingestion and labeling as rigorously as code. Actionable practices include (i) strict provenance and access control for training data sources; (ii) dataset and version signing, immutable logs, and reproducible training to enable forensic rollback [59-61]; (iii) quarantining and validating new data before it influences training; and (iv) maintaining a small, high-integrity “gold” validation set that is never influenced by untrusted channels to detect unexpected behavioral drift during retraining [61]. For poisoning-specific defenses, filtering and sanitization and robust training pipelines can help, and certified analyses have been proposed for certain classes of outlier-removal-plus-ERM defenses [62]. However, defenders should assume adaptive attackers may bypass naive sanitization, as stronger poisoning attacks have been shown to break common data sanitization defenses [63]. For backdoors in particular including supply-chain or outsourced-training scenarios, additional screening and remediation steps are warranted, such as fine-pruning to remove backdoor-related neurons with minimal accuracy loss on clean inputs [64], and runtime input-consistency tests such as STRIP-style perturbation and entropy checks to flag triggered inputs in certain modalities [65]. Taken together, these controls emphasize that protecting ML-based security systems requires securing both the model and the pipeline that produces it.

ML has become essential in modern cybersecurity as threats such as malware, phishing, and financial fraud grow increasingly sophisticated. Traditional signature-based and rule-based methods are no longer sufficient, especially against polymorphic malware, zero-day attacks, and rapidly evolving social-engineering techniques. ML enables adaptive detection by learning patterns from large datasets and analyzing behavioral, structural, and contextual features.

In malware detection, ML models, such as DT, RF, SVMs, CNNs, and hybrid CNN-LSTM architectures, identify malicious code and runtime behaviors more effectively than static approaches. Dynamic analysis further improves accuracy by evaluating malware in sandbox environments. Emerging transformer-based models show promise for capturing long-range dependencies and scaling to real-time detection.

In phishing detection, ML significantly outperforms blacklist methods by examining URL features, email content, website structure, and domain metadata. Classical models and deep learning approaches, including LSTMs and transformers, achieve high accuracy in detecting both known and zero-day phishing attacks.

In financial crime detection, ML is widely used to classify fraudulent transactions, identify anomalies, and track complex criminal networks. Techniques such as graph-based learning, deep neural networks, and even QML demonstrate strong potential.

How to cite this article:

In anomaly detection, ML models normal system and user behavior and flags deviations that may indicate intrusions or other malicious activity, including previously unseen attacks without signatures. Approaches span clustering, one-class methods such as one-class SVMs and autoencoders, and newer deep, ensemble, and graph-based techniques, but deployment must carefully balance detection sensitivity against false positive rates to avoid overwhelming analysts.

In mobile security, ML enables scalable protection for smartphones, especially Android, by detecting malicious apps and behaviors using static features, e.g., permissions, intents, API calls, code or opcode patterns, and metadata, and dynamic signals, e.g., system call traces, network flows, and sensor usage. Hybrid static and dynamic approaches and deep and sequential models improve detection of obfuscated and previously unseen malware and support real-world defenses, e.g., app-store and on-device screening, though privacy constraints, deployment overhead, and sandbox/instrumentation evasion remain key challenges.

As ML is embedded in security defenses, it also expands the attack surface: adversaries can mount evasion attacks that induce inference-time misclassification, poisoning and backdoor attacks that compromise training through corrupted data or hidden triggers, and model extraction or inference attacks that threaten confidentiality. These risks motivate defense-in-depth controls across the ML lifecycle, including robust training and evaluation, calibration and uncertainty-aware decision policies, continuous monitoring for drift and performance degradation, model and interface hardening, and end-to-end data pipeline security to prevent compromised training.

Overall, ML provides robust, adaptable defenses across all three domains, though challenges such as adversarial attacks, data imbalance, and model interpretability remain active research areas.

Ethical AI in Cybersecurity

As ML becomes increasingly integrated into cybersecurity systems, the ethical implications of AI-driven security technologies have become a critical concern. While AI enables powerful capabilities for threat detection, automated response, and large-scale monitoring, it also raises important issues related to privacy, fairness, accountability, transparency, and responsible deployment. Ethical considerations are particularly important in cybersecurity because AI systems often analyze large volumes of sensitive data and may directly influence defensive actions that affect users and organizations. Consequently, researchers and practitioners increasingly emphasize the development of trustworthy AI systems that are reliable, secure, transparent, and accountable [66].

One major ethical concern is privacy protection. Many AI-based cybersecurity systems rely on extensive datasets containing network traffic logs, user behavior traces, device metadata, and communication records. Although such data are essential for detecting threats, their use raises concerns about surveillance, unauthorized data exposure, and violations of personal privacy. Ethical deployment therefore requires privacy-preserving techniques such as data minimization, federated learning, differential privacy, and secure data governance practices [67]. These approaches aim to balance effective threat detection with protection of individual rights and regulatory compliance.

Another important issue is bias and fairness in AI models. ML systems trained on biased or incomplete datasets may generate inaccurate or discriminatory results. In cybersecurity contexts, biased models could incorrectly classify legitimate user activity as malicious or disproportionately target certain user groups or organizations. Such outcomes may lead to unfair enforcement actions, excessive monitoring, or reduced trust in automated security systems [68]. Addressing these risks requires careful dataset curation, fairness-aware learning algorithms, and ongoing auditing to detect and mitigate harmful bias in deployed models.

Transparency and explainability are also fundamental to ethical AI in cybersecurity. Many advanced detection systems rely on complex deep learning architectures that operate as “black-box” models, making it difficult for security analysts to understand why specific activities are flagged as threats [69]. Lack of interoperability can hinder incident investigation and reduce trust in automated decision-making. Explainable AI techniques aim to address this challenge by providing interpretable insights into model predictions, enabling analysts to understand, validate, and audit AI-driven security decisions.

How to cite this article:

Another key dimension is accountability and governance. When AI systems automatically detect threats or trigger defensive responses, determining responsibility for errors or unintended consequences becomes challenging. For example, automated blocking mechanisms may mistakenly interrupt legitimate services or misattribute malicious activity. Governance frameworks therefore emphasize maintaining human oversight, clear decision accountability, and detailed audit trails throughout the AI life cycle. Frameworks such as the NIST AI Risk Management Framework recommend integrating governance, risk assessment, and continuous monitoring to ensure responsible deployment of AI systems.

Ethical considerations must also account for the dual-use nature of AI technologies in cybersecurity. The same ML techniques used to defend systems can also be exploited by attackers to automate phishing campaigns, generate convincing social engineering content, or evade detection mechanisms. Recent advances in generative AI further amplify these risks by enabling large-scale automated attacks. Responsible AI research therefore requires careful disclosure policies, collaboration among academia, industry, and government, and the development of defensive strategies against AI-enabled threats [67].

To address these challenges, many organizations and researchers advocate adopting trustworthy AI principles that integrate fairness, transparency, accountability, privacy protection, and robustness throughout the AI lifecycle. These principles are reflected in emerging international standards and regulatory initiatives, including the NIST AI Risk Management Framework and other global responsible AI guidelines. Such frameworks emphasize that AI systems should be valid and reliable, safe and resilient, explainable, privacy-enhanced, and fair, thereby supporting both effective cybersecurity and ethical responsibility.

While AI significantly enhances modern cybersecurity capabilities, its deployment must be accompanied by careful consideration of ethical principles. Ensuring privacy protection, fairness, transparency, accountability, and responsible governance is essential for building trustworthy AI-based security systems. Integrating these principles into the design, training, and deployment of cybersecurity models will help ensure that AI strengthens digital security while respecting legal, ethical, and societal expectations.

Real-world applications and tools for cybersecurity

Over the past few years, ML and AI have revolutionized the field of cybersecurity. ML is now widely used in real-world applications and has become a vital technology for accurately and promptly detecting and responding to threats. ML helps make detection algorithms more efficient and accurate. It also makes the detection and mitigation faster. Many companies (Microsoft, IBM, Darktrace, CrowdStrike, Blue Hexagon, and Callsign, among many others) apply ML in their algorithms designed for malware detection, intrusion detection, fraud detection, behavioral analysis, vulnerability assessments, and bot detection, among many other use cases. As many companies have adopted AI and ML to improve their products and user experience, various cybersecurity tools have been developed to promote the information security of the company and improve digital safety for their customers. However, not all these products are widely available for public use, and some of them are made specifically for the company. In this section, we will look into real-world applications including behavioral analysis, vulnerability assessments, bot detection, and network vulnerability detection. We also take two public AI cybersecurity tools as an example to explain the practical tools for cybersecurity.

Behavioral Authentication

AI-powered behavioral analysis is becoming increasingly important among the tools in modern cybersecurity. Companies store loads of information about each user, which can be used to create a behavioral biometric for an individual user. Behavioral biometrics can capture more than two thousand different parameters about a user, for example, how long it takes the user to type, the way a user holds a phone, how fast they scroll, and even the pressure that they use to type [70]. In recent times, much of the fraud has come from using stolen identities or through malware, bots, or remote access attacks after the user has logged in [71].

ML behavioral biometrics can be used to prevent account takeover fraud. The ML algorithm takes in the different micro behaviors of a user and can identify the probability that it is the legal user or fraud [71].

How to cite this article:

CrowdStrike, a popular cybersecurity company, utilizes ML for behavioral authentication. Data scientists train the model using ground truth and known user anomalies. The ML algorithm is then given access to real-world events and user behavior data [72]. It continues to sensor and scan for anomalies and notifies the expert team when it discovers them. These anomalies are noted, action is taken, and the model is then trained again to continue to refine the detection and promote accuracy.

Vulnerability Assessments

ML can also be used to identify and prioritize vulnerabilities in a company's software systems. Vulnerability assessments analyze code and behavior to identify and rate security weaknesses. Large amounts of data related to software vulnerabilities [70] should be collected to train the ML model. After training, the ML algorithm can be applied to assess the risk of different vulnerabilities in the company's software system. It will assign each vulnerability a common vulnerability score that indicates the likelihood of the vulnerability being exploited by an attacker [70]. These common vulnerability scores help the company prioritize vulnerability levels they need to fix and in what order they should fix them. Using ML for vulnerability assessments, companies can quickly and accurately identify and assess their vulnerabilities.

Bot Detection

ML can also be utilized for bot detection [73]. A bot is an automated software program that repeatedly imitates human user behavior or performs repetitive tasks. They can be either beneficial or malicious. The malicious bots are typically used for hacking, spamming, and general compromise of computers [73]. It is estimated that almost half of the internet traffic today is made up of bots. ML can be used to help identify malware bots that often go unnoticed. As noted previously, AI-powered behavioral analysis can be used to identify bots that are imitating real users. This is one of the most powerful responses to distinguish between humans and bots. Anomaly detection is another way that ML could help identify bots. These ML algorithms can identify an unusually high number of requests from IP addresses as well as different spikes in traffic [73]. Once the ML model becomes aware of potential bots, it will automatically notify the network administrator so the bots can be dealt with in real-time. These algorithms are automated and can be scaled to identify malicious bots [73].

Network Vulnerability Detection

Network vulnerability detection is another application where ML can provide significant support. Third parties could take advantage of the network vulnerabilities in the system for unauthorized activities within the system [74]. The whole process starts with penetration testing, based on which ML is used to simulate cyberattacks to detect vulnerabilities in firewalls, networks, and overall systems. The ML-based detection uses historical data to train the model and to understand how cyberattacks occur and which ones pose big threats. Typical attacks include buffer overflow attacks, DoS attacks, backdoor attacks, and man-in-the-middle attacks [74].

ML in cybersecurity companies

ML can be used to monitor suspicious activities constantly. Cybersecurity companies are using ML to detect security issues and fix vulnerabilities before attackers exploit them, as well as to increase the accuracy and efficiency of threat detection and prevention. These companies use ML in different ways. CrowdStrike [72] uses a large amount of ML in the background to keep up with different attack vectors. The company provides real-time protection and threat intelligence.

Another company that innovatively uses ML is Callsign. Callsign heavily relies on behavioral analysis to validate a user's identity. It collects thousands of data points before using deep learning and neural networks to identify fraud, even as far as SMS phishing. The deep learning algorithm returns a confidence score for each user. The confidence score represents how sure the ML program is about whether a user is who they say they are. For example, a 0% confidence score would indicate fraud, and a 100% score would indicate the user is not a hacker. This company creates personal profiles for each user and identifies fraud on a very individual level. Another notable example is Darktrace, which leverages an AI platform on network data to identify patterns. ML models are looking for deviations from typical behavior to identify threats.

How to cite this article:

Microsoft Azure Sentinel

Cybersecurity has become an increasingly critical concern for organizations across all sizes and industries. With the evolution of cyber threats and the growing complexity of IT infrastructures, businesses are compelled to adopt modern commercial approaches to safeguard their assets and ensure normal operations. Microsoft Azure Sentinel, as a cloud-native solution, has emerged as an indispensable tool in the arsenal of cybersecurity professionals. By harnessing cutting-edge technologies such as AI and ML, the platform empowers organizations to detect, investigate, and respond to threats in real-time, thereby staying ahead of adversaries and minimizing the risk of data breaches and disruptions. The proactive and intelligent security measures reflect a fundamental evolution in the way businesses approach cybersecurity, prioritizing resilience and agility in the face of an ever-evolving threat landscape.

Microsoft Sentinel [75,76] leverages AI and ML, and equips organizations with advanced tools to proactively identify and thwart cyber threats as they evolve. Figure 1 [75] shows the architecture of the Microsoft Sentinel Security system. It provides a comprehensive view of an organization's security landscape by consolidating data from diverse sources, including users, devices, applications, and infrastructure. This holistic perspective enables organizations to stay one step ahead of potential threats, safeguarding their entire network and digital assets. One of Sentinel's key strengths lies in its intuitive interface and customizable dashboards, which empower security analysts to swiftly pinpoint suspicious activities and prioritize alerts. This agility is crucial in today's rapidly changing threat landscape, where swift responses can mean the difference between containment and a full-scale breach. Additionally, Sentinel facilitates the orchestration of automated responses, streamlining incident response processes and minimizing manual intervention. This enhances the efficiency of security operations and ensures a consistent and coordinated approach to threat mitigation across the organization.

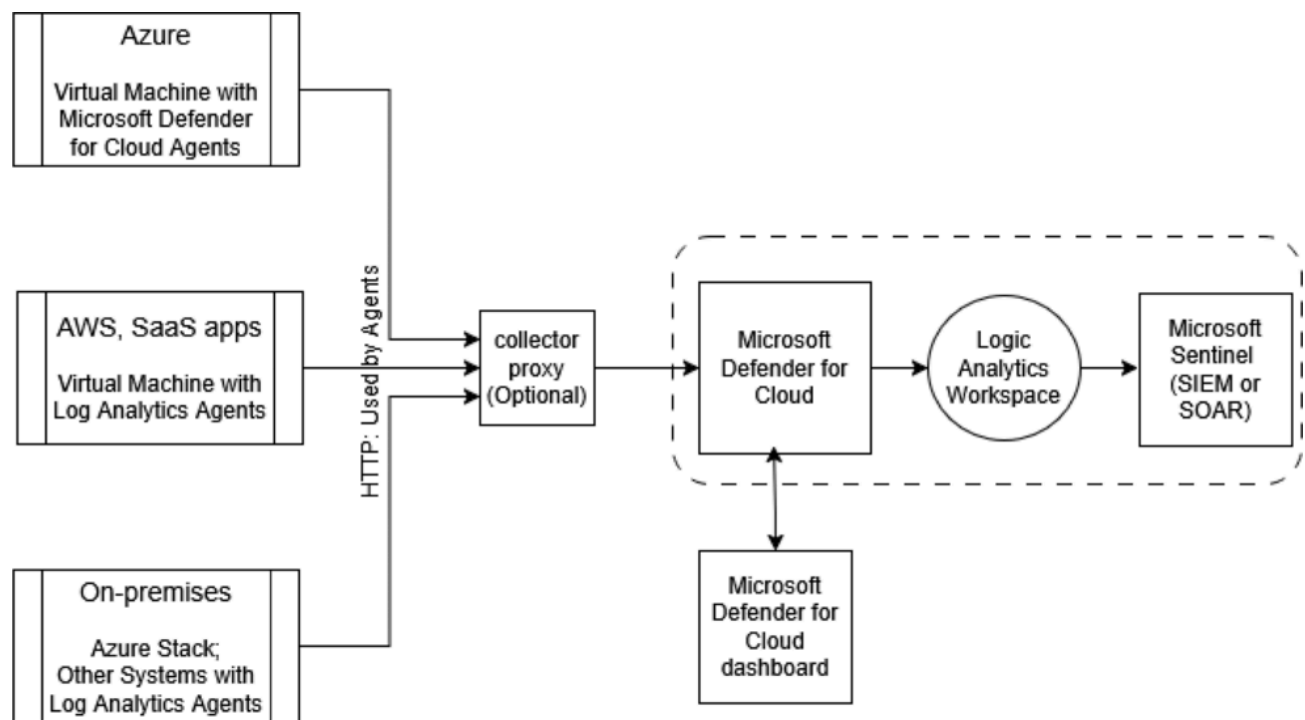


FIGURE 1: Microsoft Azure Sentinel Security

Furthermore, Sentinel serves as a proactive defense mechanism by continuously analyzing vast amounts of data to detect anomalies and potential indicators of compromise. This proactive stance enables organizations to preemptively address security weaknesses before they can be exploited by malicious actors. By offering real-time insights and

How to cite this article:

actionable intelligence, Sentinel empowers organizations to strengthen their security posture and mitigate risks effectively. Ultimately, Microsoft Sentinel plays a pivotal role in fortifying the security infrastructure of organizations, enabling them to navigate the complexities of the digital landscape with confidence and resilience.

IBM QRadar and Guardium

Another prominent technology company, IBM, has been at the forefront of leveraging AI and ML to enhance the capabilities of its cybersecurity solutions, particularly within its “QRadar” and “Guardium” [77] service suites. As IBM's flagship security intelligence platform, QRadar integrates AI and ML technologies to enable proactive threat detection and response. It analyzes vast volumes of security data in real-time, identifying patterns and anomalies indicative of potential cyber threats, which allows security teams to investigate and prioritize incidents more efficiently, thereby reducing response time and mitigating risks effectively. QRadar also provides valuable context and actionable intelligence, empowering organizations to make informed decisions to protect their digital assets.

For data security, IBM's Guardium suite harnesses AI and ML to safeguard sensitive information and ensure regulatory compliance. It continuously monitors and analyzes data access and usage patterns across heterogeneous environments, through which it can help organizations detect and mitigate insider threats, data breaches, and compliance violations. With the utilization of ML algorithms, Guardium learns and adapts to evolving data security risks, enabling proactive measures to prevent unauthorized access and data leaks. Moreover, Guardium's AI-driven insights provide organizations with a comprehensive understanding of their data landscape, facilitating informed data protection strategies and risk management practices.

The integration of AI and ML technologies into QRadar and Guardium suites represents a strategic approach to staying ahead of emerging threats and addressing the evolving demands of modern enterprises. By leveraging the power of AI for threat detection and response, and ML for data security and compliance, IBM empowers organizations to strengthen their cybersecurity posture and effectively mitigate risks in an increasingly complex and dynamic digital environment.

HyperShield by Cisco

HyperShield [78] is a recently announced security architecture by Cisco in partnership with Nvidia. It uses AI to enhance cybersecurity measures by learning from real-world cases. Released in April 2024, HyperShield has been exhibited in only a few showcases by Cisco but is considered to be a game-changer in the world of cybersecurity. It is described as using an “AI-native rule engine [that] prioritizes vulnerability and then deploys surgical compensating controls. Tested against live production traffic with optimal placement” [79]. It is trained on existing data from attacks or live attacks to predict and react to cybersecurity attacks on its own. HyperShield utilizes an extended network that continuously segments itself, which enables HyperShield to constantly adapt to new problems and create new solutions instantaneously. Cisco claims that HyperShield will be able to adapt to different situations and environments by making changes to itself, including process behaviors, file changes, and learned policy preferences. It is a breakthrough for cybersecurity as it is designed to be able to predict and respond to cyberattacks before the attack is performed.

Snyk

Founded in 2015, the Snyk company recently started incorporating AI into its product to improve security recommendations. Snyk is a publicly available AI product that tests for vulnerabilities in a plethora of user design aspects to ensure safe code. It uses its AI model to find vulnerabilities in code, open-source dependencies, container images, and cloud environments. As a top-layer product, Snyk uses ML as its backbone and aims to identify vulnerabilities before users even run their code, which is an extremely useful and revolutionary product for cybersecurity. It has multiple AI-powered functionalities, including DeepCode AI engine, AI code security fixes, AI-assisted security rules, AI code checker, Snyk Open Source, and Snyk Advisor. The DeepCode engine finds vulnerabilities and fixes them, using its AI capabilities and Snyk's one-click security fixes. These security fixes can fix security risks inside any supported IDE such as VSCode while protecting the ownership of the code. Snyk's AI functions allow users to adjust security rules based on their preferences. Snyk also provides an AI code checker that ensures the code is secure before its execution [80]. All of these

How to cite this article:

features, along with its Open-Source libraries, become a huge deal for cybersecurity. It continues to grow every day. More and more people are beginning to integrate Snyc into their development environments. With free options available to solo developers, Snyc is a very friendly option for many to introduce AI and ML into their cybersecurity development.

Conclusions

ML has emerged as a transformative force in the field of cybersecurity, offering scalable, adaptive, and intelligent solutions to a broad spectrum of cyberthreats. In this paper, we explored the theoretical foundations and practical applications of ML across key domains such as malware detection, phishing prevention, anomaly identification, financial fraud detection, mobile security, and adversarial ML. We also examined real-world tools and platforms - from behavioral authentication systems to enterprise-grade solutions like Microsoft Azure Sentinel and IBM QRadar - that showcase ML's growing impact on operational security environments. Despite its promise, the application of ML in cybersecurity is facing various challenges. Issues such as data scarcity, adversarial attacks, model interpretability, and computational demands continue to hinder widespread and reliable deployment. Moreover, the dynamic nature of cyber threats requires continuous model retraining and robust generalization capabilities. As attackers increasingly exploit AI for evasion and deception, the cybersecurity community must prioritize the development of resilient, transparent, and adaptable ML models.

Looking forward, future directions in ML for cybersecurity must prioritize robustness, transparency, and adaptability. Explainable AI is essential for improving trust and enabling security analysts and decision-makers to understand, validate, and act upon model predictions. Edge AI and lightweight learning architectures will play a critical role in enabling low-latency, privacy-preserving threat detection on mobile devices, IoT systems, and distributed networks. Federated learning offers a promising pathway to collaborative defense without centralized data sharing, addressing privacy and data availability concerns. Furthermore, emerging paradigms such as graph neural networks and quantum ML show potential for modeling complex attack relationships and accelerating large-scale threat analysis, although their practical deployment remains an open research challenge. In summary, machine learning holds immense potential to redefine cybersecurity paradigms. However, its success depends on continuous innovation, ethical application, and a strategic approach to balancing technological power with practical and secure implementation. By clearly distinguishing between established approaches and emerging research directions, this review provides both researchers and practitioners with a structured understanding of where the field stands today and how it can evolve toward more resilient, explainable, and future-proof cybersecurity systems.

Additional Information

Author Contributions

All authors have reviewed the final version to be published and agreed to be accountable for all aspects of the work.

Concept and design: Yanping Zhang, Jianjun Yang, Songqing Yue

Acquisition, analysis, or interpretation of data: Yanping Zhang, Jianjun Yang, Songqing Yue

Drafting of the manuscript: Yanping Zhang, Jianjun Yang, Songqing Yue

Critical review of the manuscript for important intellectual content: Yanping Zhang, Jianjun Yang, Songqing Yue

Supervision: Yanping Zhang

Disclosures

Conflicts of interest: In compliance with the ICMJE uniform disclosure form, all authors declare the following:

Payment/services info: All authors have declared that no financial support was received from any organization for the submitted work. **Financial relationships:** All authors have declared that they have no financial relationships at present or

How to cite this article:

Zhang Y, Yang J, Yue S (April 28, 2026) Machine Learning for Cybersecurity: Techniques, Tools, and Real-World Applications. Cureus J Comput Sci 3 : es44389-025-00059-7. DOI <https://doi.org/10.7759/s44389-025-00059-7>

within the previous three years with any organizations that might have an interest in the submitted work. **Other relationships:** All authors have declared that there are no other relationships or activities that could appear to have influenced the submitted work.

References

1. Martínez Torres J, Iglesias Comesaña C, García-Nieto PJ: [Review: machine learning techniques applied to cybersecurity](#). International Journal of Machine Learning and Cybernetics. 2019, 10:2823-2836. [10.1007/s13042-018-00906-1](#)
2. Kannappan A, Bommi RM: [An improved energy efficient chimp optimization routing algorithm for underwater acoustic sensor networks](#). 2022 International Conference on Data Science, Agents & Artificial Intelligence (ICDSAAI), Chennai, India. 2022, 1-5. [10.1109/ICDSAAI55433.2022.10028871](#)
3. Uganya G, Sudha I, Lakshmanan V, Shadrach FD, Krishnammal PM, Nandhini TJ: [Crime scene object detection from surveillance video by using tiny YOLO algorithm](#). 2023 3rd International Conference on Pervasive Computing and Social Networking (ICPCSN), Salem, India. 2023, 654-659. [10.1109/ICPCSN58827.2023.00114](#)
4. Kraiem MS, Sánchez-Hernández F, Moreno-García MN: [Selecting the suitable resampling strategy for imbalanced data classification regarding dataset properties. an approach based on association models](#). Applied Sciences. 2021, 11:8546. [10.3390/app11188546](#)
5. Belcic I, Stryker C: [What is supervised learning?](#). Accessed: April 29, 2025: <https://www.ibm.com/think/topics/supervised-learning>.
6. Akhtar MS, Feng T: [Malware analysis and detection using machine learning algorithms](#). Symmetry. 2022, 14:2304. [10.3390/sym14112304](#)
7. Zhen R, Jin Y, Hu Q, Shao Z, Nikitakos N: [Maritime anomaly detection within coastal waters based on vessel trajectory clustering and naive Bayes classifier](#). Journal of Navigation. 2017, 70:648-670. [10.1017/S0373463316000850](#)
8. Vargaftik S, Keslassy I, Orda A, Ben-Itzhak Y: [RADE: resource-efficient supervised anomaly detection using decision tree-based ensemble methods](#). Machine Learning. 2021, 110:2835-2866. [10.1007/s10994-021-06047-x](#)
9. Akhtar MS, Feng T: [Evaluation of machine learning algorithms for malware detection](#). Sensors. 2023, 23:946. [10.3390/s23020946](#)
10. Akhtar MS, Feng T: [Detection of malware by deep learning as CNN-LSTM machine learning techniques in real time](#). Symmetry. 2022, 14:2308. [10.3390/sym14112308](#)
11. Ahmad I, Wan Z, Ahmad A, Ullah SS: [A hybrid optimization model for efficient detection and classification of malware in the internet of things](#). Mathematics. 2024, 12:1437. [10.3390/math12101437](#)
12. Kapan S, Sora Gunal E: [Improved phishing attack detection with machine learning: a comprehensive evaluation of classifiers and features](#). Applied Sciences. 2023, 13:13269. [10.3390/app132413269](#)
13. Aslam S, Aslam H, Manzoor A, Chen H, Rasool A: [AntiPhishStack: lstm-based stacked generalization model for optimized phishing URL detection](#). Symmetry. 2024, 16:248. [10.3390/sym16020248](#)
14. Bezerra A, Pereira I, Rebelo MA, Coelho D, de Oliveira DA, Costa JFP, Cruz RPM: [A case study on phishing detection with a machine learning net](#). International Journal of Data Science and Analytics. 2025, 20:2001-2020. [10.1007/s41060-024-00579-w](#)
15. Aldaham SSM, Ouda O, Abd El-Aziz AA: [Improved detection of phishing websites using machine learning](#). International Journal of Intelligent Systems and Applications in Engineering. 2024, 12:4619-4633.
16. Kumar S, Ahmed R, Bharany S, et al.: [Exploitation of machine learning algorithms for detecting financial crimes based on customers' behavior](#). Sustainability. 2022, 14:13875. [10.3390/su142113875](#)
17. Darvishi H, Ciuonzo D, Eide ER, Rossi PS: [Sensor-fault detection, isolation and accommodation for digital twins via modular data-driven architecture](#). IEEE Sensors Journal. 2021, 21:4827-4838. [10.1109/JSEN.2020.3029459](#)
18. Pocher N, Zichichi M, Merizzi F, Shafiq MZ, Ferretti S: [Detecting anomalous cryptocurrency transactions: An AML/CFT application of machine learning-based forensics](#). Electronic Markets. 2023, 33:37. [10.1007/s12525-023-00654-3](#)
19. Innan N, Khan MA-Z, Bennai M: [Financial fraud detection: A comparative study of quantum machine learning models](#). International Journal of Quantum Information. 2024, 22:2350044. [10.1142/S0219749923500442](#)

How to cite this article:

20. Alhajjar E, Paul M, Nathaniel B: [Adversarial machine learning in network intrusion detection systems](#). Expert Systems with Applications. 2021, 186:115782. [10.1016/j.eswa.2021.115782](#)
21. Ferrag MA, Maglaras L, Moschogiannis S, Janicke H: [Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study](#). Journal of Information Security and Applications. 2020, 50:102419. [10.1016/j.jisa.2019.102419](#)
22. Long Z, Yan H, Shen G, Zhang X, He H, Cheng L: [A Transformer-based network intrusion detection approach for cloud security](#). Journal of Cloud Computing: Advances, Systems and Applications. 2024, 13:5. [10.1186/s13677-023-00574-9](#)
23. Koukoulis I, Syrigos I, Korakis T: [Self-supervised transformer-based contrastive learning for intrusion detection systems](#). arXiv. 2025, [10.48550/arXiv.2505.08816](#)
24. Li X, Qian C, Wang Q, et al.: [Lens: a knowledge-guided foundation model for network traffic](#). arXiv. 2024, [10.48550/arXiv.2402.03646](#)
25. Guthula S, Beltiukov R, Battula N, Guo W, Gupta A, Monga I: [Netfound: foundation model for network security](#). arXiv. 2025, [10.48550/arXiv.2310.17025](#)
26. Caville E, Lo WW, Layeghy S, Portmann M: [Anomal-E: A self-supervised network intrusion detection system based on graph neural networks](#). Knowledge-Based Systems. 2022, 258:110030. [10.1016/j.knsys.2022.110030](#)
27. Wang Y, Han Z, Du Y, Li J, He X: [BS-GAT: a network intrusion detection system based on graph neural network for edge computing](#). Cybersecurity. 2025, 8:27. [10.1186/s42400-024-00296-8](#)
28. Zhong M, Lin M, Zhang C, Xu Z: [A survey on graph neural networks for intrusion detection systems: Methods, trends and challenges](#). Computers & Security. 2024, 141:103821. [10.1016/j.cose.2024.103821](#)
29. Sattar S, Khan S, Khan MI, et al.: [Anomaly detection in encrypted network traffic using self-supervised learning](#). Scientific Reports. 2025, 15:26585. [10.1038/s41598-025-08568-0](#)
30. Singh K, Kashyap A, Cherukuri AK: [Interpretable anomaly detection in encrypted traffic using SHAP with machine learning models](#). arXiv. 2025, [10.48550/arXiv.2505.16261](#)
31. Guo H, Yuan S, Wu X: [LogBERT: log anomaly detection via BERT](#). 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China. 2021, 1-8. [10.1109/IJCNN52387.2021.9534113](#)
32. Xu J, Wu H, Wang J, Long M: [Anomaly transformer: time series anomaly detection with association discrepancy](#). arXiv. 2022, [10.48550/arXiv.2110.02642](#)
33. Tuli S, Casale G, Jennings NR: [TranAD: Deep transformer networks for anomaly detection in multivariate time series data](#). arXiv. 2022, [10.48550/arXiv.2201.07284](#)
34. Kouliaridis V, Kambourakis G: [A comprehensive survey on machine learning techniques for Android malware detection](#). Information. 2021, 12:185. [10.3390/info12050185](#)
35. Odusami M, Abayomi-Alli O, Misra S, Shobayo O, Damasevicius R, Maskeliunas R: [Android malware detection: A survey](#). Applied Informatics. ICAI 2018. Florez H, Diaz C, Chavarriaga J (ed): Springer, Cham; 2018. 942:255-266. [10.1007/978-3-030-01535-0_19](#)
36. Arp D, Spreitzenbarth M, Hübner M, Gascon H, Rieck K: [DREBIN: Effective and explainable detection of Android malware in your pocket](#). Proceedings of the Network and Distributed System Security Symposium (NDSS). 2014, 1-15.
37. Alecci M, Ruiz Jiménez PJ, Allix K, Bissyandé TF, Klein J: [AndroZoo: A retrospective with a glimpse into the future](#). MSR '24: Proceedings of the 21st International Conference on Mining Software Repositories. 2024, 389-393. [10.1145/3643991.3644863](#)
38. Gibert D, Mateu C, Planes J: [The rise of machine learning for detection and classification of malware: Research developments, trends and challenges](#). Journal of Network and Computer Applications. 2020, 153:102526. [10.1016/j.jnca.2019.102526](#)
39. Liu K, Xu S, Xu G, Zhang M, Sun D, Liu H: [A review of Android malware detection approaches based on machine learning](#). IEEE Access. 2020, 8:124579-124607. [10.1109/ACCESS.2020.3006143](#)
40. Wang X, Yiu SM: [A multi-task learning model for malware classification with useful file access pattern from API call sequence](#). arXiv. 2016, [10.48550/arXiv.1610.05945](#)
41. Shabtai A, Tenenboim-Chekina L, Mimran D, Rokach L, Shapira B, Elovici Y: [Mobile malware detection through analysis of deviations in application network behavior](#). Computer Security. 2014, 43:1-18. [10.1016/j.cose.2014.02.009](#)
42. Biggio B, Roli F: [Wild patterns: Ten years after the rise of adversarial machine learning](#). Pattern Recognition. 2018, 84:317-331. [10.1016/j.patcog.2018.07.023](#)

How to cite this article:

Zhang Y, Yang J, Yue S (April 28, 2026) Machine Learning for Cybersecurity: Techniques, Tools, and Real-World Applications. Cureus J Comput Sci : es44389-025-00059-7. DOI <https://doi.org/10.7759/s44389-025-00059-7>

43. Anderson HS, Kharkar A, Filar B, Evans D, Roth P: [Learning to evade static PE machine learning malware models via reinforcement learning](#). arXiv. 2018, [10.48550/arXiv.1801.08917](#)
44. Rigaki M, Garcia S: [Bringing a GAN to a knife-fight: adapting malware communication to avoid detection](#). 2018 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA. 2018, 70-75. [10.1109/SPW.2018.00019](#)
45. Fumera G, Pillai I, Roli F: [Spam filtering based on the analysis of text information embedded into images](#). Journal of Machine Learning Research. 2006, 7:2699-2720.
46. Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, Fergus R: [Intriguing properties of neural networks](#). arXiv, 2014. [10.48550/arXiv.1312.6199](#)
47. Li Y, Jiang Y, Li Z, Xia ST: [Backdoor learning: a survey](#). IEEE Transactions on Neural Networks and Learning Systems. 2023, 34:1119-1139. [10.1109/TNNLS.2022.3182979](#)
48. Vassilev A, Oprea A, Fordyce A, Anderson H, Davies X, Hamin M: [Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations](#), NIST Trustworthy and Responsible AI, National Institute of Standards and Technology, Gaithersburg, MD. (2025). Accessed: April 9, 2026: <https://www.nist.gov/publications/adversarial-machine-learning-taxonomy-and-terminology-attacks-and-mitigations-0>.
49. Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A: [Towards deep learning models resistant to adversarial attacks](#). arXiv. 2019, [10.48550/arXiv.1706.06083](#)
50. Zhang H, Yu Y, Jiao J, Xing EP, El Ghaoui L, Jordan M: [Theoretically principled trade-off between robustness and accuracy](#). Proceedings of the 36th International Conference on Machine Learning. 2019, 97:7472-7482.
51. Croce F, Hein M: [Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks](#). Proceedings of the 37th International Conference on Machine Learning. 2020, 119:2206-2216, 2020.
52. Cohen JM, Rosenfeld E, Kolter JZ: [Certified adversarial robustness via randomized smoothing](#). Proceedings of the 36th International Conference on Machine Learning. 2019, 97:1310-1320.
53. Guo C, Pleiss G, Sun Y, Weinberger KQ: [On calibration of modern neural networks](#). arXiv. 2017, [10.48550/arXiv.1706.04599](#)
54. Lakshminarayanan B, Pritzel A, Blundell C: [Simple and scalable predictive uncertainty estimation using deep ensembles](#). arXiv. 2017, [10.48550/arXiv.1612.01474](#)
55. Hendrycks D, Mazeika M, Dietterich T: [Deep anomaly detection with outlier exposure](#). arXiv. 2019, [10.48550/arXiv.1812.04606](#)
56. Tramer F, Zhang F, Juels A, Reiter MK, Ristenpart T: [Stealing machine learning models via prediction APIs](#). 25th USENIX Security Symposium (USENIX Security 16). 2016, 601-618.
57. [OWASP Machine Learning Security Top Ten](#). (2023). <https://owasp.org/www-project-machine-learning-security-top-10/>.
58. Shokri R, Stronati M, Song C, Shmatikov V: [Membership inference attacks against machine learning models](#). arXiv. 2017, [10.48550/arXiv.1610.05820](#)
59. Souppaya M, Scarfone K, Dodson D: [Secure Software Development Framework \(SSDF\) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities](#). NIST Special Publication 800-218. NIST, Gaithersburg, MD; 2022.
60. Spoczynski M, Melara MS, Szyller S: [Atlas: A framework for ML lifecycle provenance & transparency](#). IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Venice, Italy. 2025, 448-461. [10.1109/EuroSPW67616.2025.00058](#)
61. [Google's Secure AI Framework \(SAIF\)](#). https://safety.google/intl/en_in/safety/saif/.
62. Steinhardt J, Koh PW, Liang P: [Certified defenses for data poisoning attacks](#). NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems. 2017, 30:3520-3532.
63. Koh PW, Steinhardt J, Liang P: [Stronger data poisoning attacks break data sanitization defenses](#). Machine Learning. 2022, 111:1-47. [10.1007/s10994-021-06119-y](#)
64. Liu K, Dolan-Gavitt B, Garg S: [Fine-pruning: defending against backdooring attacks on deep neural networks](#). Research in Attacks, Intrusions, and Defenses. RAID 2018. Bailey M, Holz T, Stamatogiannakis M, Ioannidis S (ed): Springer, Cham; 2018. 273-294. [10.1007/978-3-030-00470-5_13](#)
65. Gao Y, Xu C, Wang D, Chen S, Ranasinghe DC, Nepal S: [STRIP: a defense against trojan attacks on deep neural networks](#). ACSAC '19: Proceedings of the 35th Annual Computer Security Applications Conference. 2019, 113-125. [10.1145/3359789.3359790](#)

How to cite this article:

Zhang Y, Yang J, Yue S (April 28, 2026) Machine Learning for Cybersecurity: Techniques, Tools, and Real-World Applications. Cureus J Comput Sci : es44389-025-00059-7. DOI <https://doi.org/10.7759/s44389-025-00059-7>

66. National Institute of Standards and Technology (NIST): [Artificial Intelligence Risk Management Framework \(AI RMF 1.0\)](#). NIST, Gaithersburg, MD; 2023.
67. Ajibola OS, Dopamu O, Olurin O: [Challenges and ethical implications of using AI in cybersecurity](#). International Journal of Science and Research Archive. 2025, 14:294-304. [10.30574/ijrsra.2025.14.2.0276](#)
68. [Artificial Intelligence: NIST Risk Management Framework and Guidance Addressing Bias in AI](#). <https://www.nist.gov/artificial-intelligence>.
69. Sharma A, Rani S, Shabaz M: [A comprehensive review of explainable AI in cybersecurity: Decoding the black box](#). ICT Express. 2025, 11:1200-1219. [10.1016/j.icte.2025.10.004](#)
70. Turgeman A: [Machine Learning And Behavioral Biometrics: A Match Made In Heaven](#). (2018). Accessed: April 29, 2025: <https://www.forbes.com/sites/forbestechcouncil/2018/01/18/machine-learning-and-behavioral-biometrics-a-match-made-in-...>
71. Phillips G: [These 4 Antivirus Tools Are Using AI to Protect Your System](#). (2018). Accessed: April 29, 2025: <https://www.makeuseof.com/tag/artificial-intelligence-antivirus-tools/>.
72. Stanham L: [Machine Learning \(ML\) & Cybersecurity: How is ML Used in Cybersecurity?](#). (2023). Accessed: April 29, 2025: <https://www.crowdstrike.com/en-us/cybersecurity-101/artificial-intelligence/machine-learning/>.
73. Kaspersky: [What are bots? - definition and explanation](#). Accessed: April 29, 2025: <https://usa.kaspersky.com/resource-center/definitions/what-are-bots>.
74. Industry Wired: [Future of Antivirus Protection: Top Antivirus Software Using AI](#). (2021). Accessed: April 29, 2025: <https://industrywired.com/future-of-antivirus-protection-top-antivirus-software-using-ai/>.
75. [What is Microsoft Sentinel?](#). (2024). Accessed: April 23, 2026: <https://learn.microsoft.com/en-us/azure/sentinel/sentinel-overview>.
76. [How to pitch hybrid security monitoring](#). (2025). Accessed: April 23, 2026: <https://www.alifconsulting.com/post/hybrid-security-monitoring>.
77. IBM. [Artificial intelligence \(AI\) cybersecurity solutions](#). Accessed: April 23, 2026: https://www.ibm.com/solutions/ai-cybersecurity?utm_content=SRCWW&p1=Search&p4=2446442501431&p5=e&p9=187308561792&gcls...
78. Cisco: [Unveiling a New Era of AI-native Security with Cisco Hypershield](#). (2024). Accessed: April 29, 2025: <https://newsroom.cisco.com/c/r/newsroom/en/us/a/y2024/m04/unveiling-a-new-era-of-ai-native-security-with-cisco-hypers...>
79. Cisco: [What Is Machine Learning in Security?](#). (2023). Accessed: April 29, 2025: <https://www.cisco.com/site/us/en/learn/topics/security/what-is-machine-learning-in-security.html#:~:text=In%20securit...>
80. Snyk: [The Snyk AI Security Platform](#). Accessed: April 29, 2025: https://snyk.io/product/?utm_medium=paid-search&utm_source=google&utm_campaign=gs_sn:-brand-ecpc&utm_content=br_sca&u...

How to cite this article:

Zhang Y, Yang J, Yue S (April 28, 2026) Machine Learning for Cybersecurity: Techniques, Tools, and Real-World Applications. Cureus J Comput Sci 3 : es44389-025-00059-7. DOI <https://doi.org/10.7759/s44389-025-00059-7>