



American Journal of Data Science and Artificial Intelligence (AJDSAI)

ISSN: 3069-3632 (ONLINE)

VOLUME 2 ISSUE 2 (2026)



PUBLISHED BY
E-PALLI PUBLISHERS, DELAWARE, USA

Review on Malware Detection and Classification Using Machine Learning and Deep Learning Techniques

Umesh Balami^{1*}, Bhoj Raj Ghimire¹

Article Information

Received: June 07, 2026

Accepted: August 21, 2026

Published: October 01, 2026

Keywords

*Cybersecurity, Deep Learning,
Machine Learning, Malware
Classification, Malware Detection*

ABSTRACT

Malicious software (malware) is increasing at an alarming rate, and modern malware routinely uses obfuscation techniques to evade detection, making reliable identification an open problem in cybersecurity. Signature- and heuristic-based approaches remain fast and effective against known malware but fail against unknown and zeroday variants, while behavior-based, model-checking-based, and cloud-based approaches perform better against unknown and complex threats; deep-learning-, mobile-, and IoT-based approaches have since emerged to address specific portions of the known and unknown malware landscape. This paper presents a structured literature review of malware detection approaches spanning signature-based, behavior-based, heuristic-based, model-checking-based, and deep-learning-based techniques, drawing on more than 110 primary studies published mainly between 2000 and 2020. For each approach, the underlying methodology, representative studies, reported detection performance, and documented limitations are summarized and compared. The review finds that no single approach reliably detects all malware in the wild: signature- and heuristic-based methods are fast but limited to known threats, behavior- and model-checking-based methods generalize better to unknown malware but carry higher false-positive rates and resource overhead, and deeplearning-based methods substantially reduce manual feature engineering yet remain vulnerable to adversarial evasion. These findings underscore both the difficulty of the underlying detection problem, which has been shown to be NP-complete, and the need for further research. The review concludes by motivating hybrid detection strategies that combine complementary techniques, such as behavior-based analysis with model checking or deep learning with cloud-based resources, as a promising direction for future malware detection research.

INTRODUCTION

The Internet has become central to nearly every aspect of daily life, including social interaction, banking, healthcare, and commerce, and as reliance on it has grown, so has cybercrime carried out through malicious software (malware). Malware is any software that intentionally executes malicious payloads on a target system a computer, smartphone, or network device and common categories include viruses, worms, Trojan horses, rootkits, and ransomware, each affecting its target differently, for

example by damaging the system, enabling remote code execution, or exfiltrating confidential data. Classifying malware is increasingly difficult because individual samples can exhibit characteristics of multiple classes at once. Early, traditional malware was comparatively simple and easy to detect, whereas new-generation malware runs multiple processes concurrently, can execute in kernel mode, and employs obfuscation to persist and evade detection, enabling more destructive, targeted, and persistent attacks.

Table 1: Comparison between traditional and new-generation malware

Comparison Parameter	Traditional Malware	New-Generation Malware
Implementation level	Simple coded	Hard coded
State of behavior	Static	Dynamic
Proliferation	Each copy is similar	Each copy is different
Spreading mechanism	Uses .exe extension	Uses multiple extensions
Permanence in the system	Temporal	Persistent
Interaction with processes	A few processes	Multiple processes
Use of concealment techniques	None	Yes
Attack type	General	Targeted
Defensive challenge	Easy	Difficult
Comparison Parameter	Traditional Malware	New-Generation Malware
Targeted devices	General computers	Many different devices

¹ Faculty of Science Health and Technology, Nepal Open University, Kathmandu, Nepal

* Corresponding author's e-mail: umeshbalami2015@gmail.com

The scale, sophistication, and economic cost of malware continue to grow: roughly one million new malware files are estimated to appear daily, and cybercrime was projected to cost the global economy approximately \$6 trillion annually by 2021 (Morgan, 2019). Mobile malware has risen sharply as well, with a marked increase in backdoors, fake applications, and banking Trojans targeting mobile devices (Samani & Davis, 2019), and ransomware alone was estimated to cost roughly \$11.5 billion globally by the end of 2019 (Morgan, 2019).

Malware detection is the process of determining whether a given program exhibits malicious intent. Signature-based detection was the earliest widely used approach, but it cannot reliably detect unknown or new-generation malware, which motivated behavioral, heuristic, and model-checking-based detection, alongside increasing use of data mining and machine learning. More recently, deep learning, cloud-based, mobile-device-based, and IoT-based detection approaches have been proposed. No single approach has been shown to be uniformly superior; each carries distinct advantages and disadvantages, and it is generally accepted that no algorithm can detect all malware, since malware detection has been shown to be NP-complete in multiple studies (Spinellis, 2003; Zuo *et al.*, 2005). Cohen (1987) further argued that virus detection is undecidable in the general case, and Adleman (1990) showed the problem to be intractable using Gödel numbering of partial recursive functions. In practice, new-generation malware compounds this theoretical difficulty through obfuscation techniques including encryption, oligomorphism, polymorphism, metamorphism, stealth, and packing (Szor, 2005), which allow malware to evade kernel-mode protection software such as firewalls and antivirus engines.

Against this background, this review contributes to the field by explaining new technological trends in malware creation and corresponding detection approaches, investigating the theoretical and practical limits of malware detection, summarizing the current body of malware detection studies, explaining the principal detection approaches and the methods used within each, and discussing current challenges and directions for future research. The remainder of this paper is organized as follows: the Literature Review section describes malware analysis and feature-extraction techniques, public datasets, and the five principal detection approaches together with representative related work; the Materials and Methods section states the review methodology; the Results and Discussion section compares the detection approaches, summarizes application domains, and identifies research gaps; and the Conclusions section presents the review's key findings and directions for future work.

LITERATURE REVIEW

Malware Analysis and Feature Extraction

Malware analysis determines a sample's functionality how it works, which machines and programs are affected, and what data is damaged or stolen (Sikorski & Honig,

2012). Two main techniques are used: static analysis, which examines the malware without executing it, and dynamic analysis, which examines its behavior during execution. Basic static analysis examines file strings, header information, and functions using tools such as PEiD, BinText, MD5deep, and PEview, while advanced static analysis uses a disassembler and decompiler, such as IDA Pro with Hex-Rays, to generate and interpret assembly code, though this requires strong expertise in assembly instructions and operating-system internals. Dynamic analysis executes the malware's instructions within an isolated sandbox, dedicated system, or virtual machine and examines function calls, parameters, information flows, and file, registry, and network activity; basic dynamic analysis relies on monitoring tools such as Process Monitor and Wireshark, while advanced dynamic analysis uses debuggers such as OllyDbg and WinDbg (Alosefer, 2012). In short, static analysis is faster and better suited to known malware, while dynamic analysis is more effective against unknown malware, though some variants detect sandboxed or virtualized environments and suppress their real behavior accordingly.

Extracted malware features are commonly represented using n-gram, graph-based, or vision-based models. The n-gram technique groups system calls or API calls into consecutive sequences of a specified length, but tends to generate a very large feature space that increases analysis time and can reduce model performance. In the graph-based model, system calls are converted into a graph in which nodes represent calls and edges represent relationships among them; because identifying an optimal sub-graph is NP-complete, this step is time-consuming. Vision-based feature extraction either converts malware binaries directly into an 8-bit vector represented as a 2D image, or visualizes collected execution traces (opcodes, byte strings, API and system calls) as vectors, tree maps, or graphs; prior studies have found that malware samples belonging to the same family tend to produce visually similar images (Anderson & Roth, 2018).

Relatively few malware datasets are widely accepted, publicly accessible, and formatted appropriately for data mining and machine learning. Commonly cited datasets include NSL-KDD, an updated version of the KDD'99 intrusion-detection dataset with roughly 125,000 records and 41 features (Tavallaei, 2009); Drebin, containing 5,560 Android malware samples across 20 families and 123,453 benign samples (Arp *et al.*, 2014); the Microsoft Malware Classification Challenge dataset, comprising 20,000 disassembled malware samples (Ronen *et al.*, 2018); ClAMP, with 5,184 records and 55 properties derived from PE-header API arrays; the AAGM Android malware dataset, with 400 malware and 1,500 benign samples across 12 families (Lashkari *et al.*, 2017); and EMBER, containing roughly one million records of malware and benign PE-file features (Anderson & Roth, 2018).

Malware Detection and Classification Approaches

Extracted malware features are categorized using

machine-learning algorithms or heuristic techniques to determine whether a sample is malicious. Detection and classification approaches can be grouped into five categories: signature-based, behavior-based, heuristic-based, model-checking-based, and deep-learning-based.

No single algorithm has been shown to be uniformly superior; performance depends on the specific dataset, the number and distribution of features, and dependencies among properties. Table 2 summarizes representative studies for each approach.

Table 2: Representative malware detection studies by approach

Approach	Study	Feature Representation	Reported Goal / Result
Signature-based	Schultz <i>et al.</i> (2001)	Printable strings, DLL/API calls	Detecting new malware
Signature-based	Baldangombo <i>et al.</i> (2013)	File header, DLL/API names	99% detection of known malware
Behavior-based	Wagener <i>et al.</i> (2008)	System calls, Hellinger distance	Identify new malware variants
Behavior-based	Das <i>et al.</i> (2016)	System-call frequency, n-gram	Identify new malware and variants
Heuristic-based	Anderson <i>et al.</i> (2011)	n-gram, Markov chains	96.4% detection rate
Heuristic-based	Islam <i>et al.</i> (2013)	Printable strings, API frequencies	97% detection rate
Model checking-based	Singh & Lakhoria (2003)	LTL formulas	Identify new/known malware
Model checking-based	Song & Touili (2012)	Pushdown-system stack behavior	Identify new malware
Deep learning-based	Saxe & Berlin (2015)	Contextual byte and PE-import features	95% detection, 0.1% false positives
Deep learning-based	Zhu <i>et al.</i> (2017)	Hybrid features, DBN (DeepFlow)	F1 = 95.05%

Signature-Based Detection

A signature is a malware feature that encapsulates a program's structure and uniquely identifies it. Signature-based detection is the approach most widely used within commercial antivirus software: static features are first identified from executable files, a signature-generation engine produces signatures from these features and stores them in a database, and a suspicious file is classified by comparing its extracted signature against those stored. This approach is fast and effective against known malware but fails to keep pace with new variants, is not scalable, and depends heavily on human intervention. Common signature-generation techniques include string scanning, top-and-tail scanning, entry-point scanning, and integrity checking using cryptographic hashes such as MD5 or SHA256, but none of these techniques is resistant to obfuscation, since malware can readily alter its strings and entry point.

Griffin *et al.* (2009) presented an automatic string-signature extraction method using library identification and diversity-based heuristics that reduces false-identification rates by confining generated signatures largely to malware files. Tang *et al.* (2009) applied a bioinformatics technique to generate simplified regular-expression signatures for polymorphic worms in three phases identifying prominent sub-sequences, eliminating noisy subsequences, and transforming the result into a regular expression compatible with existing intrusion-detection systems and reported the resulting schema to be noise-tolerant, though limited to polymorphic worms. Borojerdi and Abadi (2013) proposed MalHunter, which automatically generates behavioral signatures for polymorphic malware

using sequence clustering and alignment, reporting a 90.83% detection rate with a 0.80% false-positive rate, though the evaluation covered only a few hundred samples. Santos *et al.* (2009) proposed n-gram-based file signatures classified via a k-nearest-neighbor measure, demonstrating that n-gram signatures can detect unknown malware to some degree, while Zheng *et al.* (2013) presented Droid Analytics, an Android malware analysis system using a three-level signature scheme argued to be more resistant to packing and mutation than traditional hash-based signatures.

Overall, signature-based detection remains fast and effective for malware belonging to a known family, but fails against new-generation malware using obfuscation and polymorphism, is prone to false positives, and requires labor-intensive signature extraction. Building a more effective signature-based scheme would benefit from shorter signatures that still cover many variants, more robust automatic generation, greater use of data mining and machine learning during generation, and stronger resistance to packing and obfuscation.

Behavior-Based Detection

Behavior-based detection observes a program's behavior using monitoring tools to decide whether it is malicious or benign. Because a program's behavior tends to remain similar even as its code changes, this approach can detect a majority of new malware variants, though some malware behaves atypically inside a protected environment and may be mistakenly classified as benign. The approach comprises three stages: extracting behaviors (via automated sandbox analysis, or by monitoring system

calls, file changes, registry snapshots, network activity, or processes), deriving properties from them, and classifying the sample using machine-learning algorithms.

Kolbitsch *et al.* (2009) proposed a graph-based detection method representing system calls as nodes and transitions as edges, comparing a sample's call graph against known malicious and benign graphs; the method performs well on known malware but struggles with unknown malware, since it observes only partial execution paths. Lanzi *et al.* (2010) proposed a system-centric behavioral model (AccessMiner) that compares a sample's interaction with system resources against known malicious and benign behavior groups, detecting a significant fraction of malware with few false positives, though it cannot detect malware that avoids hiding its presence or gaining OS control. Wagener *et al.* (2008) proposed a flexible, automated technique that extracts behaviors from system calls and computes Hellinger distance between them, reporting detection of obfuscated variants with similar behaviors, though without a statistical performance evaluation against prior work. Chandramohan *et al.* (2013) proposed Bounded Feature Space Behavior Modeling, which substantially reduces the number of properties by converting interrelated system calls into higher-level behaviors before classification, offering fixed dimensionality and therefore efficiency and scalability, though it ignores system-call frequency. Pajouh *et al.* (2018) used a kernelbased support vector machine with library-call-frequency weighting to detect Mac OS X malware, reporting a 91% detection rate with a 3.9% false-positive rate and noting that increasing sample size improved accuracy while reducing false positives.

Behavior-based detection generally proceeds through determining behaviors, extracting features from them, and applying classification, using techniques such as n-gram, n-tuple, bag, and graph models to derive features and distance or similarity measures such as Hellinger distance and cosine coefficient to compare them. The difficulty of precisely defining behaviors, the large feature spaces produced by n-gram-style methods, and the tendency of some malware to behave atypically inside virtual machines or sandboxes have limited the effectiveness of behavior-based systems, though continued integration of machine learning and data mining for meaningful feature generation remains an active research area. More recently, multivariate time-series techniques such as vector autoregression and dynamic Bayesian networks have been applied to capture temporal dependencies among network metrics, improving detection of anomalies such as denial-of-service attacks, port scanning, and data exfiltration relative to traditional univariate approaches (Howard *et al.*, 2026).

Heuristic-Based Detection

Heuristic-based detection is a composite approach that combines experience-derived rules with machinelearning techniques, drawing on both string- and behavior-related features to generate rules from which signatures are created. It is used primarily to identify variants of known

malware as well as some previously unseen samples, and although its success rate against new malware is comparatively high, both false-positive and false-negative rates also tend to be high due to optimization trade-offs. Anderson *et al.* (2011) proposed graph-based detection using dynamically collected instruction-trace graphs, Markov chains to estimate transition probabilities, and support-vector-machine classification over a graphkernel similarity matrix, reporting a 96.41% detection rate and significant improvement over signature-based methods. Islam *et al.* (2013) combined static and dynamic features method-length frequencies, printable strings, and systemcall parameters into a single feature vector, reporting a 97% detection rate, though the detection rate for unknown malware remained comparatively lower and the false-positive rate comparatively higher than desired. Ye *et al.* (2010) proposed post-processing techniques rule pruning, ranking, and selection for associative-classificationbased malware detection, reporting performance exceeding commercial antivirus tools as well as naive Bayes, support-vector-machine, and decision-tree baselines. Naval *et al.* (2015) proposed a dynamic detection system that constructs a graph of system calls and searches for semantically relevant paths, reporting a higher detection rate for system-call injection attacks than comparable methods, though the search for all relevant paths is itself NPcomplete and the method carries computational overhead.

Heuristic-based detection combines string and behavioral features to generate rules and, from them, signatures, drawing on API calls, control-flow graphs, n-grams, opcodes, and hybrid features (Bazrafshan *et al.*, 2013). While it can detect various forms of known and unknown malware, it remains insufficient against the newest generation of malware and is prone to relatively high false-positive rates.

Model Checking-Based Detection

Model-checking-based detection extracts malicious and benign features and encodes them as linear temporal logic formulas capturing feature dependencies, comparing a sample's extracted properties against previously defined specifications. This approach is resistant to stealth and packing techniques and can detect a portion of new malware variants. Kinder *et al.* (2005) proposed a proactive, signature-update-free model-checking method that extracts control-flow graphs from executables and validates them against computation tree predicate logic (CTPL) specifications, reporting detection of various worm forms with a low false-positive rate, though the approach requires manual steps and omits data-flow analysis. Song and Touili (2012, 2014) proposed pushdown-modelchecking methods that reduce the model-checking problem to controlling a Büchi pushdown system with symbolic variables, reporting resistance to stealth techniques and high detection accuracy, though the method works correctly only when stack data cannot be modified via direct memory access. Battista *et al.* (2016)

applied model checking to Android malware, analyzing Java bytecode to detect the DroidKungFu and Opfake families with reportedly high accuracy, though the study covered only a few malware families.

Model checking is a mature technique for program verification but has not yet been used extensively for malware detection. While effective against some new malware variants, it remains insufficient against all complex malware and is comparatively resource-intensive, yielding only a limited view of overall malware behavior. Improving the accuracy of extracted specifications and using more effective logic formulas are promising directions for further work.

Deep Learning-Based Detection

Deep learning is a subfield of machine learning that inherits from artificial neural networks and learns directly from examples. While widely adopted in image processing, autonomous vehicles, and voice control, it has seen comparatively less use in malware detection. Deep-learning-based detection achieves high performance and substantially reduces the required feature dimension, but is not resistant to evasion attacks, and building additional hidden layers is time-consuming while tending to yield only marginal performance improvements.

Saxe and Berlin (2015) proposed a deep-neural-network detection system using two-dimensional software features extracting complementary feature types, building a network with two hidden layers, and applying a calibrator score to the output reporting a 95% detection rate with a low false-positive rate under crossvalidation, though performance dropped sharply under split validation. Dahl *et al.* (2013) proposed large-scale malware classification using random projections and neural networks, reducing input dimensionality by a factor of 45 and training on over 2.6 million labeled samples, reporting a two-class error rate of 0.49% for a single network and 0.42% for an ensemble. Yuan *et al.* (2014) proposed Droid-Sec, combining static and dynamic analysis with a deep-belief-network pre-training phase followed by supervised fine-tuning, reporting 96% accuracy that outperformed support-vector-machine, decision-tree, logistic-regression, and naive-Bayes baselines. Huang and Stokes (2016) proposed MtNet, a multi-task neural-network architecture for dynamic malware classification trained on 4.5 million files, reporting substantial improvement over a shallow neural architecture, though additional layers yielded little further improvement and the model remained susceptible to evasion. Zhu *et al.* (2017) proposed DeepFlow, combining hybrid features with a deep-belief-network model, reporting an F1 score of 95.05%, and Ye *et al.* (2018) proposed DeepAM, a heterogeneous deep-learning framework using multi-layer constrained Boltzmann machines with associative memory, reporting improved performance relative to traditional shallow-learning methods for zero-day malware detection. Deep-learning-based detection nonetheless remains vulnerable to adversarial evasion. Grosse *et al.* (2016) investigated adversarial crafting against deep neural networks using the Drebin dataset, achieving

misclassification rates of up to 80%, and Kolosnjaji *et al.* (2018) demonstrated a gradient-based attack capable of evading a raw-byte-based deep network by modifying fewer than 1% of a sample's bytes while preserving its malicious functionality. In summary, deep-learning-based detection is comparatively new to the field: it is powerful and substantially reduces the required feature space, but remains vulnerable to evasion attacks, and further empirical study is needed to characterize its strengths and limitations more precisely.

MATERIALS AND METHODS

This review adopts a structured, narrative literature-review methodology. Primary studies were drawn mainly from peer-reviewed conference proceedings, journal articles, and technical reports published between 2000 and 2020, retrieved from IEEE Xplore, ACM Digital Library, Springer, ScienceDirect, and related indexing services using search terms combining “malware detection”, “malware classification”, “signature-based”, “behaviorbased”, “heuristic-based”, “model checking”, “machine learning”, and “deep learning”. More than 110 primary studies met the inclusion criteria of (a) proposing or evaluating a malware detection or classification method, (b) reporting a clearly identifiable methodology and, where available, a quantitative performance measure, and (c) being published in a peer-reviewed venue or a widely cited technical report.

For each retained study, five attributes were extracted: the detection approach category (signature-, behavior-, heuristic-, model-checking-, or deep-learning-based), the feature-representation technique used, the reported detection goal and performance (accuracy, precision, recall, or false-positive rate, where available), the dataset or malware corpus used, and the limitations documented by the original authors. Studies were then grouped by detection approach and synthesized qualitatively, since the heterogeneity of datasets, metrics, and experimental protocols across studies precluded a direct meta-analytic pooling of results. This categorization scheme, summarized in Table 2, and the qualitative and pros/cons comparisons presented in the Results and Discussion section (Table 3, Table 4, Figure 1, and Figure 2) follow directly from this extraction process.

RESULTS AND DISCUSSION

Reported Detection Performance

Reported accuracy figures vary considerably across sub-domains, feature representations, and evaluation datasets, and are not directly comparable given differing malware corpora and validation protocols. Figure 1 charts a representative selection of accuracy figures reported in the reviewed literature between 2011 and 2020, illustrating the general upward trend in reported accuracy as hybrid and deep-learning-based models became more common, particularly for Android and IoT-focused sub-domains; most selected studies cluster between 95% and 99.7% accuracy, with the lowest reported figure (91.4%) coming from an earlier hybrid DBN model evaluated in 2015.

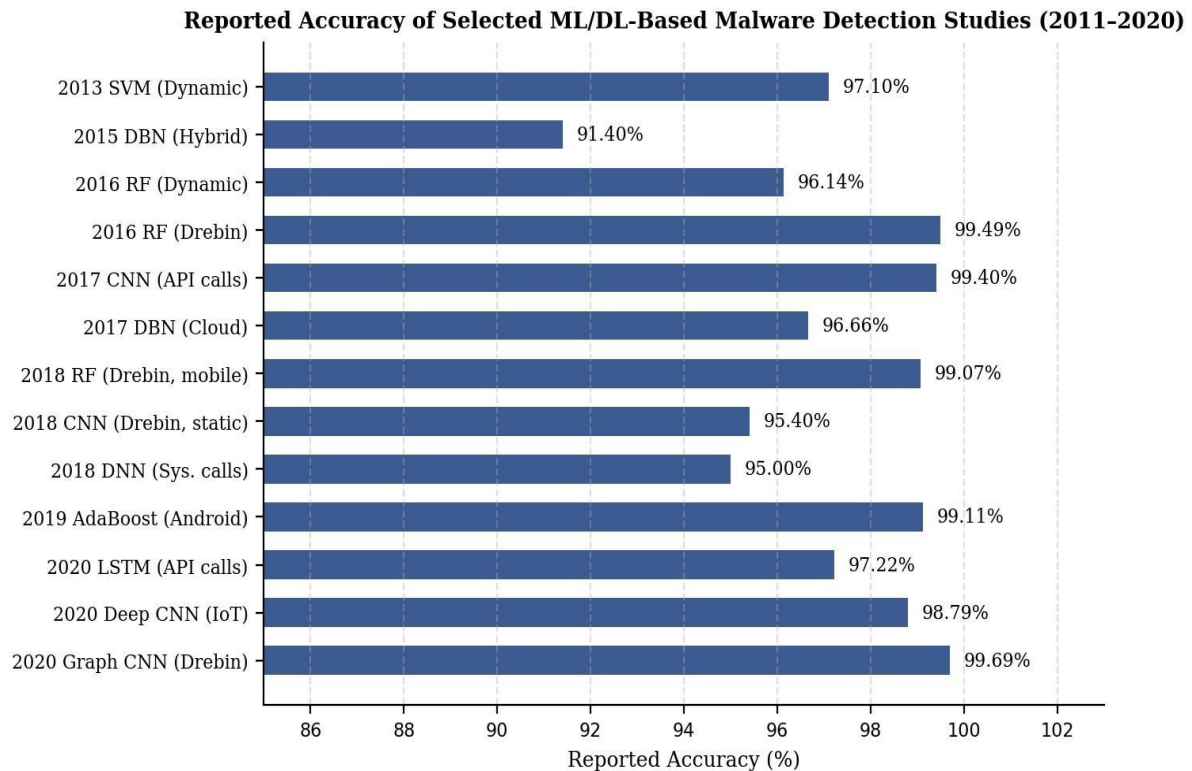


Figure 1: Reported accuracy of selected machine learning and deep learning-based malware detection studies (2011-2020), representative selection

Comparative Evaluation of Detection Approaches

Signature-, behavior-, heuristic-, and model-checking-based approaches are well established and have been used for malware detection for more than a decade, drawing on reverse engineering, data mining, and machine-learning techniques. Signature-based detection is fast and effective against known malware using static features such as byte sequences, assembly instructions, strings, opcodes, and DLL lists, but cannot detect new-generation malware and is vulnerable to obfuscation and polymorphism. Behavior-based detection determines a program's functionality, which tends to persist even as instruction sequences and signatures change, so it can detect new malware and variants of the same family and is comparatively effective against obfuscation and polymorphism; however, it produces a high rate of false positives, and some malware behaves differently or fails to run at all in a protected environment, causing it to be mistakenly marked benign.

Heuristic-based detection can use both static and dynamic features and can detect some previously unknown malware, but remains vulnerable to metamorphic techniques, and the large number of rules and extensive training required make this approach comparatively complex. Model-checking-based detection is powerful, can detect unknown malware, and resists obfuscation and polymorphism, but yields only a limited view of overall malware behavior and is not resistant to evasion attacks. Deep-learning-based approaches are effective at detecting new malware and sharply reduce the required feature space, but remain vulnerable to certain evasion attacks, and in cloud deployments, overhead between client and server and the lack of continuous real-time monitoring remain open challenges. Despite each method's particular advantages for specific datasets, no single detection method can currently detect all malware, as summarized in Table 3, Table 4, and Figure 2.

Table 3: Pros and cons of each malware detection approach

Detection Approach	Advantages	Limitations
Signature-based	Fast and highly efficient against known malware; widely used in commercial antivirus software.	Fails against unknown or zero-day threats; highly vulnerable to code obfuscation.
Behavior-based	Maps the underlying functionality of a program; effective against unknown and obfuscated variants.	High false-positive rate; malware may alter behavior inside a sandbox or virtual machine.
Detection Approach	Advantages	Limitations

Heuristic-based	Can discover previously unseen variants by combining static and dynamic traits into rules.	Requires complex training; vulnerable to metamorphic changes and higher error rates.
Model checkingbased	Mathematically verifies code against known malicious traits; robust against obfuscation.	Resource-intensive; yields only a limited view of overall malware behavior.
Deep learning-based	Drastically reduces the required feature space; learns features automatically.	Vulnerable to adversarial evasion; training additional hidden layers is time-consuming.

Figure 2 visualizes two of the qualitative criteria from Table 4 side by side detection of unknown malware and resistance to obfuscation highlighting that model-checking-based detection is the only approach rated favorably on both dimensions, while signature-based detection is rated favorably on neither, and behavior-, heuristic-, and deeplearning-based approaches each trade one dimension off against the other.

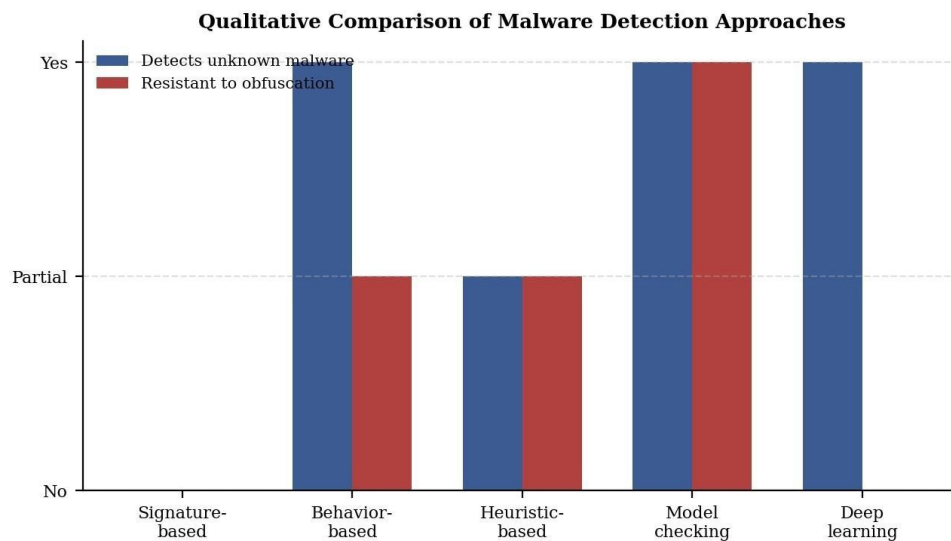


Figure 2: Qualitative comparison of malware detection approaches (detects unknown malware vs. resistant to obfuscation)

Table 4: Comparison of malware detection approaches by qualitative criteria

Detection Approach	Detects Unknown Malware	Resistant to Obfuscation	Well-Established	Emerging Approach
Signature-based	No	No	Yes	No
Behavior-based	Yes	Partial	Yes	No
Heuristic-based	Partial	Partial	Yes	No
Model checking-based	Yes	Yes	Partial	Partial
Deep learning-based	Yes	No	No	Yes

Application Areas

The findings of this review are applicable to the deployment of malware detection methods across several digital environments and asset classes, including cybersecurity defense systems that enhance organizational, governmental, and private-sector infrastructure; enterprise and corporate security, protecting networks, servers, and endpoints; mobile-device security, monitoring smartphones and tablets through API-call analysis and behavior profiling; embedded detection within antivirus programs and intrusion-detection systems for real-time protection; threat-intelligence and incident-response platforms informed by behavior-based and model-checking insights; and research and

development of next-generation detection tools that combine multiple approaches, such as deep learning with cloud-based analysis, to address evolving threats. In the financial sector specifically, AI-based anomaly detection techniques have been shown to strengthen safeguards against fraud and cyber theft, though their adoption remains constrained by low technical know-how, limited integration of explainable AI, and the high cost of procurement and maintenance, particularly in developing economies (Chinonye & Onah, 2025).

Research Gaps

This review identifies several notable gaps in the current malware detection landscape. Existing approaches,

particularly signature-based methods, struggle against sophisticated and obfuscated malware, and while behavior-, model-checking-, and deep-learning-based methods offer improvements, none comprehensively detects all unknown or highly sophisticated variants. The absence of well-accepted, comprehensive, and standardized datasets impedes objective benchmarking and comparison across detection methods. Many approaches that perform well on controlled datasets remain difficult to deploy for continuous, real-time monitoring in dynamic environments such as IoT and mobile ecosystems, and no current approach detects every form of malware present in real-world settings, with unknown, zero-day, and polymorphic malware remaining especially elusive. Most detection methods also suffer from high false-alarm rates, particularly when malicious and benign software share similar features, reducing both user trust and practical applicability; developing models resilient to obfuscation, packing, and adversarial evasion remains an open research area.

CONCLUSION

Although numerous methods have been proposed across the five detection approaches reviewed in this paper, no single method can detect all new-generation and sophisticated malware. Signature- and heuristic-based approaches perform well for known malware, while behavior- and model-checking-based approaches perform better against unknown and complex malware, and deep-learning-based approaches have emerged to address a further portion of both known and unknown malware, though a meaningful share of malware still evades all current approaches. Building an effective, general-purpose detection method therefore remains a substantial challenge, and there is a clear opportunity for further research, particularly hybrid approaches that combine complementary techniques, such as behavior-based detection with model checking, or deep learning with cloudbased analysis, to leverage the specific strengths of each and improve detection of unknown and obfuscated malware. As malware volume, severity, and sophistication continue to rise, and the associated economic cost continues to grow, timely detection before damage occurs remains essential for individuals, companies, and governments alike, and continued integration of data mining, machine learning, and deep learning into hybrid, multi-modal detection frameworks represents the most promising direction for future work in this field.

REFERENCES

- Adleman, L. M. (1990). An abstract theory of computer viruses. In *Advances in Cryptology—CRYPTO* (pp. 354–374). Springer-Verlag.
- Alosefer, Y. (2012). *Analysing web-based malware behaviour through client honeypots* [Doctoral dissertation, Cardiff University]. <https://orca.cardiff.ac.uk/id/eprint/29469/>
- Anderson, B., Quist, D., Neil, J., Storlie, C., & Lane, T. (2011). Graph-based malware detection using dynamic analysis. *Journal of Computer Virology*, 7(4), 247–258.
- Anderson, S., & Roth, P. (2018). *EMBER: An open dataset for training static PE malware machine learning models*. arXiv:1804.04637.
- Arp, D., Spreitzenbarth, M., Hübner, M., Gascon, H., & Rieck, K. (2014). Drebin: Effective and explainable detection of Android malware in your pocket. In *Proceedings of the Network and Distributed System Security Symposium* (Vol. 14, pp. 23–26).
- Battista, P., Mercaldo, F., Nardone, V., Santone, A., & Visaggio, C. A. (2016). Identification of Android malware families with model checking. In *Proceedings of the 2nd International Conference on Information Systems Security and Privacy*.
- Bazrafshan, Z., Hashemi, H., Fard, S. M. H., & Hamzeh, A. (2013). A survey on heuristic malware detection techniques. In *Proceedings of the 5th Conference on Information and Knowledge Technology*.
- Borojerdi, H. R., & Abadi, M. (2013). MalHunter: Automatic generation of multiple behavioral signatures for polymorphic malware detection. In *Proceedings of the International Conference on Computer and Knowledge Engineering (ICCKE)* (Vol. 1). Ferdowsi University of Mashhad.
- Chandramohan, M., Tan, H. B. K., Briand, L. C., Shar, L. K., & Padmanabhuni, B. M. (2013). A scalable approach for malware detection through bounded feature space behavior modeling. In *Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering* (pp. 312–322).
- Chinonye, A. E., & Onah, B. I. (2025). Assessing the efficacy of AI-based techniques in anomaly detection in financial institutions. *American Journal of Data Science and Artificial Intelligence*, 1(2), 11–17. <https://doi.org/10.54536/ajdsai.v1i2.5062>
- Cohen, F. (1987). Computer viruses: Theory and experiments. *Computers & Security*, 6(1), 22–35.
- Dahl, G. E., Stokes, J. W., Deng, L., & Yu, D. (2013). Large-scale malware classification using random projections and neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*.
- Griffin, K., Schneider, S., Hu, X., & Chiueh, T.-C. (2009). Automatic generation of string signatures for malware detection. In *Proceedings of the International Workshop on Recent Advances in Intrusion Detection*. Springer.
- Grosse, K., Papernot, N., Manoharan, P., Backes, M., & McDaniel, P. (2016). *Adversarial perturbations against deep neural networks for malware classification*. arXiv:1606.04435.
- Howard, C. C., Otobo, F. N., & Odogu, E. (2026). Analyzing temporal dependency structures in cyber security: A case study of network traffic anomalies. *American Journal of Data Science and Artificial Intelligence*, 2(1), 22–32. <https://doi.org/10.54536/ajdsai.v2i1.5081>

- Huang, W., & Stokes, J. W. (2016). MtNet: A multi-task neural network for dynamic malware classification. In *Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer.
- Islam, R., Tian, R., Batten, L. M., & Versteeg, S. (2013). Classification of malware based on integrated static and dynamic features. *Journal of Network and Computer Applications*, 36(2), 646–656.
- Kinder, J., Katzenbeisser, S., Schallhart, C., & Veith, H. (2005). Detecting malicious code by model checking. In *Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer.
- Kolbitsch, C., Comparetti, P. M., Kruegel, C., Kirda, E., Zhou, X.-Y., & Wang, X. (2009). Effective and efficient malware detection at the end host. In *Proceedings of the USENIX Security Symposium* (Vol. 4, No. 1, pp. 351–366).
- Kolosnjaji, B., Demontis, A., Biggio, B., Maiorca, D., Giacinto, G., Eckert, C., & Roli, F. (2018). Adversarial malware binaries: Evading deep learning for malware detection in executables. In *Proceedings of the 26th European Signal Processing Conference (EUSIPCO)*.
- Lanzi, A., Balzarotti, D., Kruegel, C., Christodorescu, M., & Kirda, E. (2010). AccessMiner: Using system-centric models for malware protection. In *Proceedings of the 17th ACM Conference on Computer and Communications Security* (pp. 399–412).
- Lashkari, A. H., Kadir, A. F. A., Gonzalez, H., Mbah, K. F., & Ghorbani, A. A. (2017). Towards a network-based framework for Android malware detection and characterization. In *Proceedings of the 15th Annual Conference on Privacy, Security and Trust (PST)*.
- Morgan, S. (2019). *2019 cybersecurity almanac: 100 facts, figures, predictions and statistics*. Cisco and Cybersecurity Ventures. <https://cybersecurityventures.com/cybersecurity-almanac-2019>
- Naval, S., Laxmi, V., Rajarajan, M., Gaur, M. S., & Conti, M. (2015). Employing program semantics for malware detection. *IEEE Transactions on Information Forensics and Security*, 10(12), 2591–2604.
- Pajouh, H. H., Dehghantanha, A., Khayami, R., & Choo, K.-K. R. (2018). Intelligent OS X malware threat detection with code inspection. *Journal of Computer Virology and Hacking Techniques*, 14(3), 213–223.
- Ronen, R., Radu, M., Feuerstein, C., Yom-Tov, E., & Ahmadi, M. (2018). *Microsoft malware classification challenge*. arXiv. <https://arxiv.org/abs/1802.10135>
- Samani, R., & Davis, G. (2019). *McAfee mobile threat report: Q1 2019*. McAfee. <https://www.mcafee.com/enterprise/en-us/assets/reports/rp-mobile-threat-report-2019.pdf>
- Santos, I., Penya, Y. K., Devesa, J., & Bringas, P. G. (2009). N-grams-based file signatures for malware detection. In *Proceedings of the 11th International Conference on Enterprise Information Systems* (Vol. 9, pp. 317–320).
- Saxe, J., & Berlin, K. (2015). Deep neural network based malware detection using two dimensional binary program features. In *Proceedings of the 10th International Conference on Malicious and Unwanted Software (MALWARE)*.
- Sikorski, M., & Honig, A. (2012). *Practical malware analysis: The hands-on guide to dissecting malicious software*. No Starch Press.
- Song, F., & Touili, T. (2012). Efficient malware detection using model-checking. In *Proceedings of the International Symposium on Formal Techniques*. Springer.
- Song, F., & Touili, T. (2014). Pushdown model checking for malware detection. *International Journal on Software Tools for Technology Transfer*, 16(2), 147–173.
- Spinellis, D. (2003). Reliable identification of bounded-length viruses is NP-complete. *IEEE Transactions on Information Theory*, 49(1), 280–284.
- Szor, P. (2005). *The art of computer virus research and defense*. Pearson Education.
- Tang, Y., Xiao, B., & Lu, X. (2009). Using a bioinformatics approach to generate accurate exploit-based signatures for polymorphic worms. *Computers & Security*, 28(8), 827–842.
- Tavallae, M. (2009). A detailed analysis of the KDD CUP 99 data set. In *Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications* (pp. 1–6).
- Wagener, G., State, R., & Dulaunoy, A. (2008). Malware behaviour analysis. *Journal in Computer Virology*, 4(4), 279–287.
- Ye, Y., Li, T., Jiang, Q., & Wang, Y. (2010). CIMDS: Adapting postprocessing techniques of associative classification for malware detection. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 40(3), 298–307.
- Ye, Y., Chen, L., Hou, S., Hardy, W., & Li, X. (2018). DeepAM: A heterogeneous deep learning framework for intelligent malware detection. *Knowledge and Information Systems*, 54(2), 265–285.
- Yuan, Z., Lu, Y., Wang, Z., & Xue, Y. (2014). Droid-Sec: Deep learning in Android malware detection. *ACM SIGCOMM Computer Communication Review*, 44(4), 371–372.
- Zheng, M., Sun, M., & Lui, J. C. (2013). Droid analytics: A signature based analytic system to collect, extract, analyze and associate Android malware. In *Proceedings of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*.
- Zhu, D., Jin, H., Yang, Y., Wu, D., & Chen, W. (2017). DeepFlow: Deep learning-based malware detection by mining Android application for abnormal usage of sensitive data. In *Proceedings of the IEEE Symposium on Computers and Communications (ISCC)*.
- Zuo, Z., Zhu, Q., & Zhou, M. (2005). On the time complexity of computer viruses. *IEEE Transactions on Information Theory*, 51(8), 2962–2966.