International Journal on Optimization and Applications

IJOA. Vol. 05, Issue No. 03, Year 2025, www.ijoa-journal.com

Copyright © 2025 by **I**nternational **J**ournal on **O**ptimization and **A**pplications

Enhancing Malware Anomaly Detection through Static Analysis, Machine Learning, and Similarity Hashing

Mina Zouhal
mina.zouhal@uit.ac.ma
ENSA kenitra

Khalid Chougdali
khalid.chougdali@uit.ac.ma
ENSA kenitra

khalid Bennouk
khalid.bennouk@acgcybersecurity.fr

Dorra Mahouachi
dorra.mahouachi@acgcybersecurity.fr

Abstract. This paper presents a hybrid malware detection method that combines machine learning, fuzzy hashing, and static analysis. The method uses similarity-based hashing to find possible variants after extracting structural and semantic characteristics from executable files, such as metadata and cryptographic signatures. Models for classification that can differentiate between dangerous and benign software are trained using these enriched features. Static analysis and clever algorithms together improve resilience against obfuscated malware while maintaining accuracy in real-world detection tasks, according to experimental evaluation.

I. Introduction

Endpoint Detection and Response (EDR) solutions are the linchpin of cyber security, providing ongoing monitoring and analysis of endpoint activity that not even the most sophisticated attackers can evade. They have evolved to become more advanced at data gathering, behavioral analysis and automated response. But the massive amount and intricacy of endpoint data often make separating out unwanted behavior from the unwanted nearly impossible without the aid of smarter analytic methods. Detecting malware on endpoints and doing so efficiently remains a key challenge with rapidly changing threat landscape. The complexity of modern assaults highlights the shortcomings of traditional static signature-based techniques. A multilayer detection approach that combines sophisticated static analysis, machine learning's predictive power, and hash comparison's dependability shows promise as a means of boosting system security. Static analysis allows for the detection of questionable patterns and structures without execution, whereas machine learning offers the capacity to recognize abnormalities and unknown risks. Lastly, hashing guarantees the quick detection of malware that has already been cataloged. The goal of this tactical combination is to greatly strengthen EDRs' security posture against present and potential threats. Machine learning (ML) has been recently proposed as a promising approach to overcome such limitations, by enabling the derivation of complex patterns from large collections of data and the identification of weak signals of malicious activities. Varying supervised and unsupervised learning methods, including decision trees, support vector machines, and deep learning models, have been utilized to find anomalies, classify malware, and forecast threats. While the approaches of this kind greatly elevated the detection precision level, they come with the corresponding liabilities of the type of excessive false positive rates or uninterpretability that make it impossible for the security analysts to comprehend the rationale of the warning. Graph analysis can aid in focusing investigation and detecting complex patterns of harmful behaviors. However, generating and analysing large-scale, dynamic graphs from endpoint data presents major computational and scalability issues. Evolution of EDR Techniques. As cybersecurity threats became more sophisticated, EDR solutions underwent fundamental modifications.

Copyright © 2025 by **I**nternational **J**ournal on **O**ptimization and **A**pplications

Table 1. Progression from early reactive approaches to current proactive defense paradigms :

Years	Techniques Used in EDR
2005-2010	Signatured based Detection, Heuristic Detection, Log analysis, host-based intrusion detection systems (HIDS), Firewall and Antivirus Software, Manual Incident Response, Disk Imaging and Forensics,Endpoint Encryptions, Policy-based Control.
2011-2015	Behavioural Analysis, Memory Forensics, Network Traffic Analysis, Indicators of Compromise (IoC). Detection, Heuristic Detections, Sandboxing, Anomaly Detection.
2016-2020	Machine Learning and Behaviour Analytics, Endpoint Isolation, Cloud based EDR solutions, File less Malware Detections, Deception Technologies, IoT and OT Endpoint Security, User and Entity Behaviour Analytics (UEBA), (Application and Programming Interface) API Integration and Thread Intelligence Sharing.
2021-2023	Extended Detection and Response (XDR), Zero Trust Architecture, Artificial Intelligence (AI) powered Thread Detection[7][8], Threat Intelligence-driven Defence, Behavioural Biometrics, Cloud-native EDR solutions, Automated Incident response, Ecosystem Integration.

II. Related Work

The article**[2]** « **Investigating Proactive Digital Forensics Leveraging Adversary Emulation (**2022**)** » of '**Valentine Machaka 1 and Titus Balan**'This literature review defines digital forensics, distinguishes proactive, active, and reactive approaches, and identifies proactive forensics as a means of saving on expenditure and boosting efficiency through preparation of the system for laterinvestigations well in advance. Proactive forensics, the review details, involves the installation of policies, procedures, tools, and technologies for collecting, warehousing, and reviewing evidence ; it also covers related domains such as computerintrusion forensics and forensic readiness in a focus on the use of audit trails and logging ; and it coversintrusion detection technologies (HIDS, NIDS, and EDR) and evidence acquisition use. Also noted are the limitations of the classical forensic approaches in the contemporary system, and the necessity for ahead-of-time preparation for digital forensics, including data integrity and safe communication protection, particularly in the system of the cloud where security and forensics intersect. Proactive digital forensics' conceptual architecture takes on a Security Operations Center (SOC) continuously monitoring endpoints and networks with infrastructure in a state ready for swift incident response and evidence availability. Experimental scenario isolates a social engineering assault with a malicious MicrosoftExcel document containing a Covenant C2 framework payload. On opening up the document and executing the macros, alerts were raised, suggesting a machine-under-the-influence and a suspicious PowerShellfile download request via HTTP. Analysts investigated these suspicions, utilizing tools such as CapME to gather a collection of network traffic data and understand the malicious PowerShellscript block. Such proactive step necessitates verification against sophisticated threats - a gap bridged in later studies. Building on forensic readiness, the work **Fagbohunmi Grifin Siji, Okafor Patrick Uche** proposed[3] this document « **An improved model for comparing different endpoint detection and response tools for mitigating insider threat (**2023**)**» of The manuscript offers a superior mathematical modeling for determining the relative effectiveness of Endpoint Protection Platforms (EPP) and Endpoint Detection and Response (EDR) solutions in mitigating menace, particularly insider threats. It determines the merits and weaknesses of each platform and provides for selection of the finest protection for organizations of various scales on the basis of specific situations of operation. MATLAB simulation and modeling outcomes demonstrate that EPP is especially effective in loss minimization in cases where threats cause prolonged system inactivity via initial compromise prevention. It is inversely disadvantaged in using machine learning approaches for detection of newly emergent or previously unknown threats but lags behind in tracking all endpoints and promptly responding to threats propagated via the network. Such modeling provides insightful contributions towards customized security strategy consideration for account of menace properties, expenditure, in addition to platform possibilities and acts as a significant contribution towards organizations in security platform selection. yet simultaneously uncover operating shortcomings that are particularly taxing on sophisticated assault vectors**.**This document [4] « **AnEmpirical Assessment of Endpoint Detection and Response Systems against Advanced Persistent Threats Attack Vectors(**2021**) »of «George Karantzas 1 and Constantinos Patsakis »** . The study performs empirical testing of EDRs against APT attacks through a number of various simulated attack

Copyright © 2025 by **I**nternational **J**ournal on **O**ptimization and **A**pplications

situations in controlled networks to observe the performance in such circumstances. It establishes that, despite technological advances, most APT attacks are not deterred or sufficiently reported by the platforms, exposing significant weaknesses in their ability to confront such sophisticated attacks. The paper also examines the various means malicious actors use to evade detection, such as obfuscation and interfering with collection mechanisms, in a way that they are able to breach security controls and move on to more clandestinely operate. Additionally, the study establishes that there exist available telemetry mechanisms, presenting an additional challenge to detection and allowing attacks more difficult to observe. It establishes that, in their current form, most EDRs are not stopping or sufficiently reporting most attacks, and significant upgrades are necessary to make them more effective. Lastly, the study establishes the importance of boosting the detection capabilities, logging and resilience of EDR platforms in an attempt to counteract evasion tactics, and informs future updates and strategic placement of such tools in businesses.This evidence of EDR vulnerabilities against sophisticated attacks necessitates more advanced detection methods, particularly for detecting lateral movement.[5] **« Detecting lateral movement: A systematic survey** »of « **Christos Smiliotopoulosa, Georgios Kambourakisa, Constantinos Kolias(2023)** » It details the increased importance of lateral movement (LM) detection methodologies in today's network security, particularly through the application of Endpoint Detection and Response (EDR) technologies and machine learning technologies, namely neural networks (NN). Several studied works are grounded on log- driven EDR solutions tracking real-time endpoint activity in real-time for identifying anomalous or malicious behavior corresponding to stages of attacks for early and specific detection of privilege escalation or pivoting attacks. Further, deep neural networks (DNN), recurrent models including LSTM, or attention- related models (GAN) are necessary to express and identify complex patterns of anomalous activity in vast amounts of data. Such machine learning technologies extend detection accuracy, especially in IoT or extended IT (IIoT) networks where methods of attacks are more sophisticated and varied. Therefore, the integration of log- driven EDR methodologies and advanced neural network technologies is a promising solution for lateral movement detection challenges, resulting in more reactive and responsive protection mechanisms against sophisticated attacks including APTs or LM. though the heavy reliance on log analysis introduces new challenges regarding data integrity and manipulation detection.[6] These challenges are directly tackled by **Markus Wurzenberger** et al **« Analysis of statistical properties of variables in log data for advanced anomaly detection in cyber security (2021)** » Variable Type Detector (VTD) introduced in the present work offers a significant contribution towards detecting anomalies in log data, particularly for Endpoint Detection and Response (EDR) solutions. It identifies variable type transitions signifying maliciousity through independent examination of variable portions in logs and thereby boosts the real-time detection capability in EDR solutions. Among the notable achievements of VTD is the autonomous analysis of all the log data without preliminary knowledge or human curation, enabling streamlined implementation in a heterogeneous infrastructure. Moreover, the addition of VTD in currently implemented architectures such as SIEM, EDR, and XDR enriches the level of automation while reducing false positivity, optimizing the functions of remediation and response in cybersecurity. Briefly, VTD offers a solution towards a novel fusion of statistical reasoning and automation towards boosting the efficiency of EDR system in monitoring endpoint log data. Such a feature enables advanced threat detection proactivity and brings security resilience in the infrastructure system as a whole.

Benchmark

To assess the effectiveness of modern incident detection and response (EDR) systems, this study compiles insights from recent research and benchmarking analyses. Current approaches show a wide diversity, including :

- Adversary emulation for proactive forensic investigation
- Dynamic access control models to mitigate insider threats,
- Statistical log analysis and empirical testing

Against advanced persistent threats (APTs). Benchmark results reveal critical tradeoffs : no solution fully covers the MITRE ATT & CK architecture, although several achieve a high level of automation and a low false positive rate. Moreover, there are significant differences in performance overhead, particularly for resource -intensive methods such as machine learning - based behavioral analysis or sandboxing. The need for more portable,

Copyright © 2025 by International Journal on Optimization and Applications

understandable, and thorough detection methods is highlighted by these gaps.

Table 2. Comparative cybersecurity detection techniques performance analysis

Article	Technique Focus	Coverage (ATT&CK %)	Performance Impact	False Positives	Detection Precision
Machaka & Balan (2022) – Proactive Forensics	Adversary emulation, DFIR tools	~75%	Moderate	Low	~85%
Siji & Uche (2023) – Insider Threat Mitigation	Mathematical modeling, access mg.	~60%	Moderate-High	Medium	~80%
Karantzas & Patsakis (2021) – EDR vs APT	Sandbox + ML + signature	~65%	High	Medium-High	~70%
Smiliotopoulos et al. (2023) – Lateral Movement Detection	Log analysis + DNN models	~80%	Moderate	Low	~88%
Wurzenberger et al. (2021) – Statistical Log Analysis	Variable Type Detector (VTD)	~78%	Low-Moderate	Low	~90%

III. Anomaly Detection Methods

Malware can be divided into several categories depending on the attackers ' objectives . For example, ransomware encrypts the victim's system to prevent access and demands payment to decrypt files , while spyware is designed to discreetly monitor user actions and steal sensitive information such as passwords or banking details . Cyber criminals are often motivated by the desire to make money, gather industrial or political intelligence , or disrupt the economy and politics . Table 1.1 summarizes the main categories of malware and the most common use of each.

Malware infection vectors are open to modification depending on the targeted operating system. Some popular points of intrusion are PDF files (.pdf), Office suite files such as Word (.doc/.docx), Excel (.xls/.xlsx), PowerPoint (.ppt/.pptx), and even executable files. Executables can exist in many forms : ELF files in Linux-based systems, APK files in Android-based systems, as well as Portable Executables (PE) in Microsoft Windows-based systems. In this study, we focus primarily on the detection and analysis of executable malware samples (e.g., ELF/PE files, implants, RATs) since Windows remains the most widely used operating system worldwide. Besides the type of files used, the attackers utilize an array of attack strategies to achieve a range of malicious objectives such as financial blackmailing, spying, and system

Copyright © 2025 by International Journal on Optimization and Applications

disruption. These attack forms include phishing attacks, infections with ransomware, denial-of-service (DDoS) attacks, penetrations of networks, and password attacks. Table 1.2 is an explanation of the attacks and matches them with the most common malicious intentions. Effective detection of malware at the endpoint is a high priority in the face of relentlessly evolving cyber threats. The advanced level of the newest attacks underlines the weakness of detection by static signatures alone. A multi-level detection approach—combining advanced static analysis with the predictive capability of machine learning and the consistency of hash comparison—has great promise in bolstering the security of systems. While machine learning has the promise of discovering anomalies and unknown threats previously unknown, static analysis has the ability of discovering suspicious patterns and structures without execution of the actual file itself. Not least of all, the hash-based methods promise rapid detection of recognized kinds of malware. Such a strategic combination aims at significantly enhancing the security posture of Endpoint Detection and Response (EDR) systems against existing and future threats.

Table 3.Attack Types vs Strategic Cyber Threat Objectives

Type of Attack	Financial Extortion	Espionage	Destabilisation
Phishing	X	X	X
Ransomware	X		
Denial-of-Service (DDoS)			X
Network Intrusion		X	X
Malware (General)	X	X	X
Password Attacks	X	X	

IV. Anomaly and Threat Detection Model in Computer Security

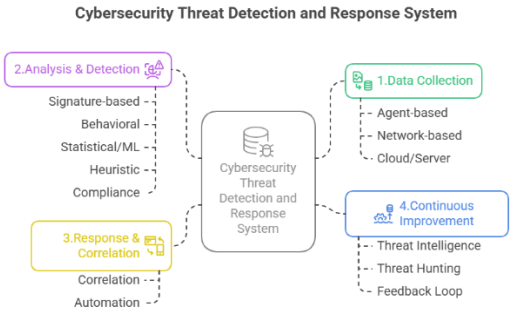

Fig 1. Anomalies detections

Data Collection : The first step is data collection, limited here to endpoint agents and network systems. Windows agents on endpoints collect logs from systems, such as Sysmon or event logs, while Linux agents retrieve kernel traces like eBPF and system activity information. Information is gathered from network devices, including firewalls that capture traffic logs, login attempts, and security alarms. This limited gathering still provides useful visibility by combining perspectives from the host level and the network level, to provide a reliable starting point for centralized correlation and analysis.

Analysis & Detection : In this step , the collected data is analyzed to detect potential threats . Signature-based analysis checks for known patterns using antivirus databases , YARA rules , or IDS rules , allowing malware to be quickly spotted using hashes , domain names, or IP addresses . Behavioral detection focuses on unusual actions like bizarre process chains , persistence attempts, or suspicious use of system tools , often associated with frameworks such as MITRE ATT&CK. Statistical analysis and machine learning add another dimension, allowing for the detection of anomalies , grouping similar behaviors , or classifying files and events based on technical characteristics like headers , imports , or entropy . Heuristics and scoring allow for assigning a risk level to each event to help prioritize the most critical alerts . In our work , we combine static analysis , hashing , and artificial intelligence: static analysis helps extract useful features from files , hashing (including fuzzy hashing) helps identify malware families and variants, and artificial intelligence helps detect threats . unknown or evasive .

Response & Correlation : After detection, the system moves to response and correlation, which aim to understand the attack and stop it quickly . Correlation links events based on time , entity (such as a user, host, or process), or cause, allowing

Copyright © 2025 by International Journal on Optimization and Applications

analysts to view the entire attack chain rather than isolated alerts . This reduces noise and groups alerts into meaningful incidents . Automated response via SOAR platforms allows for rapid action , such as isolating an infected endpoint, blocking a checksum , IP address , or domain , containing a malicious email , or resetting compromised accounts . Human analysts remain involved for high - impact decisions, with Tier 1 personnel performing triage and Tier 2 personnel handling escalations . The success of this step is measured by indicators such as mean time to detect (MTTD), mean time to respond (MTTR) and the reduction of false positives.

Continuous Improvement : Since cyber threats are constantly evolving, the system must continuously improve. Every alert and incident provides valuable feedback : false positives and false negatives spotted by analysts are used to retrain machine learning models, adjust thresholds , and refine detection rules . Threat intelligence feeds provide new indicators and techniques used by attackers, which can be used for proactive threat hunting or retrospective research in historical data . Security teams also conduct simulations such as red teaming, purple teaming , and ATT&CK - based exercises to test and validate defenses . Post - incident reviews identify lessons learned, which are then used to improve detection and response capabilities. These continuous improvements ensure that defenses remain effective and able to evolve in the face of new attack techniques.

Feedback Loop : The final step is the feedback loop, which ensures improvements are applied throughout the system. Analysis results, detection effectiveness, and response results are used to update detection playbooks, refine collection strategies, and improve enrichment and machine learning models . Performance metrics like mean time to detect (MTTD), mean time to respond (MTTR), attacker technique coverage, and cost per alert indicate where to focus improvements and demonstrate the overall value of the system.

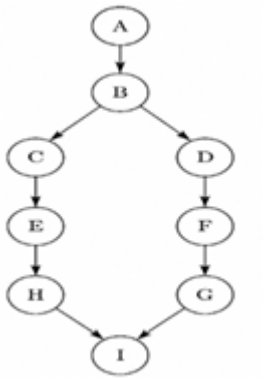

Fig 2. Control flow graph (CFG) G=(V,E) For static analysis

This cycle allows cybersecurity defenses to improve and become more effective over time, providing dynamic and adaptive protection against future cyberattacks.

V. Methodologie

a. Static analysis

Here, we narrow our focus to consider static analysis, which is a form of detailed analysis that entails studying a malware program without running it. Using this method, we are able to look at the internal configuration of the malware, extract its fundamental parts, and ascertain what it might do simply by looking at the artifacts present. Static analysis requires the scrutinizing of metadata, file sections, and pertinent signatures, as well as looking for strange strings, imported functions, and specific commands.

b. Theoretical foundations

Formal theory of language uses Chomsky-type grammars to study languages that are extremely syntactically disciplined. Syntactic investigation, or parsing, checks that code obeys these rules by applying automata and context-sensitive grammars. Finite automata, often built from regular expressions to locate specific patterns, are used to match regular languages, for example in malware detection. Static analysis applies advanced methods such as polyhedral models to compute loop invariants without relying on heuristics. The concise nature of regular expressions allows patterns to be matched in both text and code. Finally, program semantics, independent of syntax, focuses on behavior and meaning.

c. Control Flow Graph (CFG) Definition

A Control Flow Graph **(CFG)** of a program **P** is defined as a directed graph **G(P)=(V,E,entry,exit)** , where V is the set of basic blocks (maximal sequences of instructions without internal jumps) and $E \subseteq V \times V$ represents possible transfers of control. The nodes **entry** and **exit** denote the starting and ending blocks of the program. An edge **$(b,b') \in E$** exists if execution can move from the last instruction of block **b** to the first of block **b′.** A control-flow path is a sequence of connected blocks from **entry** to **exit.** By construction, every concrete program execution corresponds to a path in **G(P),** ensuring that the CFG faithfully represents all possible flows of control in the program.

Copyright © 2025 by **I**nternational **J**ournal on **O**ptimization and **A**pplications

G=(V,E)

V={v1,v2,...,vn}: set of nodes.

E={(vi,vj)| a transition is possible from vi to vj}: set of edges.

Definitions for Data Flow Analysis

Two sets, **Def(n)** and **Use(n),** are defined for every node nnn in the control flow graph **(CFG)** in order to examine data flow. The variables that are defined or assigned a value at node n are contained in the set **Def(n),** whereas the variables that are read or used at that same node are contained in the set **Use(n).** These definitions are essential to several static analysis methods, including def-use chain creation, reaching definitions, and live variable analysis, which all aid in locating potentially harmful activity or redundant and dead code in executable systems.

d. Machine Learning for Malware Classification

The growing number and complexity of malware samples makes traditional analysis methods increasingly challenging. To address this problem. Machine learning offers a scalable and automated method for detecting and classifying malware. By learning patterns and representations specific to malicious behavior, machine learning algorithms can detect structural and behavioral clues from large sets of executable files. This study examines the use of supervised and unsupervised learning techniques to classify malware into known families based on representations of binary and disassembled files.

Dataset and Initial Processing

The dataset used in this work consists of labeled malware samples provided in two complementary formats :

1. **Hexadecimal bytecode representations (. Bytes files)**
2. **Disassembled assembly code (.asm files)**

Each sample is uniquely identified and linked to a specific malware family, as defined in the provided trainLabels.csv file. This file contains manually assigned labels corresponding to nine distinct malware families , numerically coded from 1 to 9. These labels are loaded into a Pandas DataFrame, creating a reference framework for the entire analysis. This organized association between the raw data (bytecode and assembler) and its semantic labels (malware family) is essential to ensure consistency during feature extraction, transformation , and classification . The structure of the dataset thus ensures that each observation is correctly associated with its ground truth throughout the modeling process.

Supervised Learning Models[7]

XGBoost (Extreme Gradient Boosting)

XGBoost is an ensemble learning algorithm based on gradient boosting, which constructs additive decision trees to minimize a loss function augmented by a regularization term. It is particularly suited for structured data and is known for its high performance in classification tasks.

- **Objective function** :
  $L(\theta)=\sum l(y_i, y'_i) + \sum \Omega(f_k)$
- **Update rule (Newton-Raphson method)** :
  $W_{t+1}=w_t-(\nabla^2 L)-1\nabla L$
- **Split gain formula** :

$$\text{Gain}=\frac{(\sum gi)2}{\sum hi+\lambda}$$

Where **gi** and **hi** are first and second-order derivatives of the loss function, respectively.

Random Forest

Random Forests are ensemble methods that average predictions from multiple decision trees built on bootstrapped datasets. Each tree uses a randomly selected subset of features, which introduces decorrelation and improves generalization.

- **Splitting criterion**

  Gini index $=1-\sum p_i^2$, where pip_ipi is the proportion of class iii at a node.

- **Feature selection** :

At each split, only a subset of features is considered, which promotes diversity among trees and stabilizes predictions.

Unsupervised Learning Model[8]

Forest Insulation

While XGBoost and Random Forest are supervised methods , Isolation Forest is an unsupervised algorithm designed specifically for anomaly detection . It isolates anomalies (e.g., new or obfuscated malware) by randomly selecting features and values , building trees in

Copyright © 2025 by International Journal on Optimization and Applications

which anomalies are more likely to appear near the root . The main idea is that malicious samples often show behaviors that differ greatly from normal or common malware , making them simpler to identify . This makes Isolation Tree useful for detecting unknown or zero - day malware .

e. Hashing, Fuzzy Hashing, and Hamming Distance in Malware Detection

Une fonction de hachage transforme une entrée x en une sortie de taille fixe h(x) , souvent appelée valeur de hachage ou digest. Les fonctions de hachage cryptographiques classiques, comme MD5 et SHA-256, sont conçues pour être résistantes aux collisions, ce qui signifie qu'il est impossible, du moins de manière computationally viable, de trouver deux entrées différentes x_1 et x_2 telles que $h(x_1) = h(x_2)$. Ces caractéristiques les rendent très utilisés dans les systèmes d'indexation, de vérification de l'intégrité et de détection basée sur les signatures.

In the proposed Endpoint Detection and Response (EDR) architecture, a lightweight agent monitors endpoint activities, collecting information about files , processes, network connections , and system events . Each file is processed by a hashing engine (e.g., **MD5, SHA-256, or TLSH**). This allows for rapid comparison of unknown files with databases containing known malware (blacklists) or verified safe files (whitelists). Suspicious files undergo additional static analysis (e.g. , searching for strings , binary sections , imports) as well as dynamic analysis (e.g. , sandboxing , API call monitoring) . The extracted features are then represented as vectors and sent to machine learning models for classification.

While traditional hashing is efficient , it is binary in nature : two files are either identical (same hash) or completely different (different hash). This limitation makes it unsuitable for combating polymorphic and metabolic malware , as slight code changes result in entirely different hashes . To address this, we introduce fuzzy hashing and Hamming distance as complementary techniques.

Fuzzy hashing [9](e.g., ssdeep, TLSH) generates digests that preserve similarity, allowing for approximate file matching . Instead of requiring exact equality, fuzzy hashes can measure the similarity between two inputs, making them particularly useful against obfuscated or slightly modified malware. **The Hamming distance** [10] indicates how many bits differ between two binary strings of the same length. In malware detection, this can be applied to fuzzy hash outputs or binary representations of extracted signatures. A low Hamming distance means high similarity, suggesting that the new file might be a variant or obfuscated form of a known malware sample.

Why is a low Hamming distance suspicious ?

Malware variants : Attackers often release new versions of malware with minor code modifications . These modifications maintain a high similarity to the original example, resulting in a small Hamming distance.

Obfuscation techniques :[11] Code obfuscation (e.g., reordering instructions , inserting unnecessary code , encrypting code blocks) changes the appearance of the file while maintaining its functionality. Despite these modifications , the obfuscated file remains close to the original in terms of bit-level structure.

Polymorphic malware :[12] More advanced software constantly modifies its code. However, certain structural or behavioral patterns persist, allowing their detection through similarity analysis based on Hamming distance, particularly when focusing on key features rather than entire binaries.

By combining fuzzy hashing and Hamming distance in signature - based detection, the EDR system becomes more resilient to code transformations and malware evolution, thus bridging the gap between exact signature matching and fully behavior - based detection.

Our solution is organized into three main steps : signature matching using fuzzy hashing, static and behavioral feature extraction , and intelligent decision making using machine learning . Unlike traditional cryptographic hashes (like MD5 or SHA-256), fuzzy hashing produces digests that maintain similarity between files , allowing the system to compare files that are not identical but share similar structures . This makes it particularly effective for detecting obfuscated or slightly modified malware . Basic structural similarity can be identified using a low Hamming distance. Poor polymorphic and shapeshifting malware : Although more sophisticated, these types of malware alter their code with each occurrence. However, some parts of the code or behaviors may remain identical, allowing for Hamming distance - based detection, especially if the scan focuses on key features rather than the entire.

Copyright © 2025 by **I**nternational **J**ournal on **O**ptimization and **A**pplications

To further refine this process, we incorporate the Hamming distance algorithm to calculate similarity scores between fuzzy hash digests. By comparing the number of bits that differ between a new sample's signature and that of known malware , the system can spot closely related variants or disguised threats.

Raw security data must undergo a thorough preparation step before it can be analyzed. By creating comparable signatures for files with common characteristics , SSDEEP fuzzy hashing [14] plays a key role in this situation. This can effectively identify partial similarities that usual

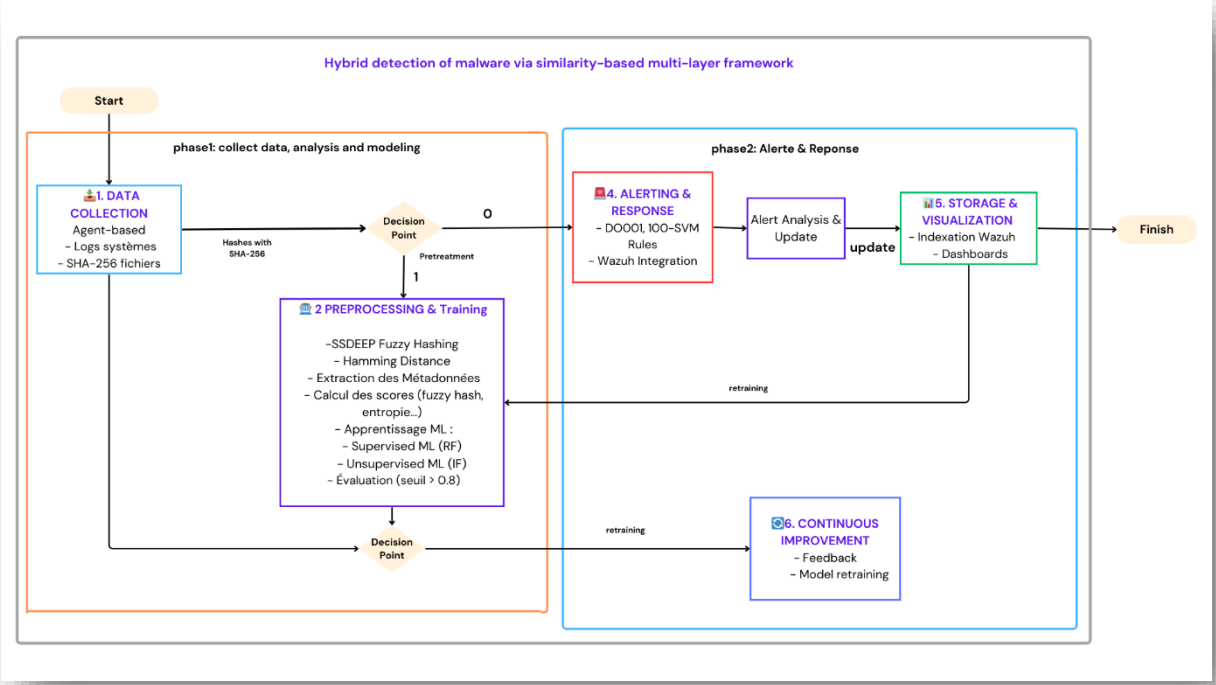

Fig 3.Malware Detection and Analysis Flow

A low Hamming distance indicates a high similarity, suggesting that the file could be a modified version of an existing malware sample.

Phase 1: Collect Data, Analysis, and Modeling

- ***Step 1: Data Collection***

The hybrid detection methodology begins with a comprehensive collection of data for analysis. Agents deployed on target systems capture system logs in real time and calculate **SHA-256**[13] fingerprints of all suspicious files. This first step builds a first line of protection by detecting known files. The system then employs SSDEEP fuzzy hashing to find even partial similarities across files, augmented with Hamming distance calculation and metadata extraction. Before any further processing decisions are taken, these techniques allow for multidimensional examination of dubious files.

- ***Step 2: Preprocessing***

cryptographic hashes fail to detect due to minor modifications. By dividing files into blocks , hashing each block , and creating a composite signature for comparison , SSDEEP achieves this . To identify file variations , including modified malware samples , it generates a similarity score between 0 and 100. These drawbacks include difficulties with extremely small files or heavily restructured data , although it works well for code reuse or small modifications . However, SSDEEP is still frequently used in digital forensics , malware analysis, and cybersecurity.

Hamming distance[15] is used to precisely measure the bit - by - bit differences between file signatures, as well as for fuzzy hashing. By comparing their bit patterns, this method is particularly useful for detecting malicious files that have been slightly modified. For example , a Hamming distance of 1 means a close match with known malware when comparing " 101010 " and "101000". This is where this method excels by finding perfect or near- exact matches . Furthermore , by providing important context such as timestamps , file attributes , and

Copyright © 2025 by **I**nternational **J**ournal on **O**ptimization and **A**pplications

behavior patterns , extracting metadata from system logs enhances the study. Data normalization ensures that each feature contributes equally to the subsequent analysis , which helps better prepare the dataset for effective model training . Finally, feature engineering transforms these raw observations into useful variables for machine learning models.

Table 4. Bit Position & Signature

Bit Position	Signature A	Signature B	Difference
1	1	1	0
2	0	0	0
3	1	1	0
4	0	0	0
5	1	0	1
6	0	0	0

Table 5.
Hash-based file classification rules

Condition	Action	Decision Code
SHA-256 or fuzzy hash present in the whitelist	Safe file, no need for ML	0
SHA-256 or fuzzy hash present in the blacklist, or fuzzy hash > 90%	Known malware, immediate alert	0
Fuzzy hash between 20% and 80%	Gray zone → proceed to ML	1
Fuzzy hash < 20%, not present in blacklist or whitelist	Very different → potentially legitimate or unknown → ML can help	1
Fuzzy hash > 80% and < 90%, not present in the blacklist (upper intermediate zone)	ML can help confirm	1

- ***Step 3 : Training the Model***

Both supervised and unsupervised methods are used in the machine learning component. In order to find known threat patterns, Random Forest classifiers learn from labeled malware/benign samples. Isolation Forest algorithms find unusual file properties to uncover new risks. File hashes, information, and behavioral characteristics are among the historical data used to train the models. To maintain detection accuracy, fresh threat intelligence is included into regular retraining cycles. Before models are deployed, performance measurements make sure they fulfill minimum detection thresholds.

Decision Point 1 : Initial Filtering

The system applies initial filtering rules before engaging machine learning. Files matching whitelist entries are immediately cleared as safe. Known malicious files matching blacklist signatures trigger immediate alerts. Files with fuzzy hash similarity scores above 90% to known malware are blocked automatically. Only files in the "gray zone" (20-80% similarity) proceed to ML analysis. This tiered approach optimizes resource usage by reserving ML for ambiguous cases.

Second Decision Point: ML Assessment

Multiple criteria are used by trained models to assess suspicious files. Files with a confidence level greater than 0.8 are marked as malicious and set off response processes. Additional manual evaluation is performed for detections with medium-confidence (0.5-0.8). Files with low confidence are either made public or given more thorough examination. For auditing purposes, all decisions are recorded together with supporting documentation. To increase the accuracy of future detections, the system adds all evaluation results to its knowledge base.

Phase 2: Alerts & Response

The response workflow begins when malware detections are verified. To stop the spread, automated containment procedures isolate impacted systems. Multiple sources of contextual data are used to enhance incident specifics. Prioritization of alerts is determined by the probable impact and the seriousness of the threat. Security teams follow

Copyright © 2025 by **I**nternational **J**ournal on **O**ptimization and **A**pplications

response playbooks as they conduct investigations and corrective actions. Every action is recorded for process improvement and compliance reporting.

- ***Step 4: Alerting & Response***

The alerting system integrates with existing security infrastructure. Wazuh correlation rules trigger notifications based on detection confidence levels. Automated responses may include process termination, file quarantine, or network isolation. Security teams receive enriched alerts with threat intelligence context. Response timelines are tracked to ensure SLA compliance. Feedback loops capture analyst assessments to refine future detections.

- ***Step 5: Visualization & Storage***

A centralized data lake houses all security occurrences. Wazuh indexing makes it possible to quickly search through security data from the past. Threat trends and detection statistics are displayed on custom dashboards. Framework performance measures and areas for improvement are highlighted in reports. Policies for data retention strike a balance between storage limitations and forensic requirements. Appropriate data visibility across teams is ensured by role-based access controls.

- ***Step 6: Continuous Improvement***

Feedback methods are incorporated into the architecture to facilitate continuous improvement. To find gaps in detection, false positives and negatives are examined. Retraining of the model takes place after significant events and at predetermined intervals. Signature databases are updated every day by threat intelligence feeds. Response times and detection rates are monitored using performance metrics. Periodically, the entire system is subjected to security reviews in order to handle new threats.

IV. RESULTS AND DISCUSSION

Precision, Recall, F1-score, and Accuracy are common classification metrics that were used to assess the performance of the implemented models. The global performance for each model is shown in Table 2, while the specific per-class performance (Benign vs. Malware) is summarized in Table 1.

Table 6 .Classification report by class

Model	Class	Precision	Recall	F1-score	Support
XGBoost	0 (Benign)	0.97	0.99	0.98	2282
	1 (Malware)	0.99	0.98	0.99	2998
Random Forest	0 (Benign)	0.97	0.99	0.98	2282
	1 (Malware)	0.99	0.98	0.99	2998
Isolation Forest	0 (Benign)	0.42	0.87	0.56	2282
	1 (Malware)	0.42	0.07	0.13	2998

Table 7. Global model performance

Model	Accuracy	Precision	Recall	F1-score
XGBoost	0.9835	0.9929	0.9780	0.9854
Random Forest	0.9848	0.9949	0.9783	0.9865
Isolation Forest	0.4170	0.4242	0.0747	0.1271

Analysis

The findings show that, with an overall accuracy of over 98%, XGBoost and Random Forest performed almost identically. Both models had F1-scores near 0.99 for both the benign and malware classes, demonstrating exceptional precision (>0.99) and recall (~0.98). This suggests that when trained on characteristics obtained from cryptographic hashes, fuzzy hashing (ssdeep, TLSH), and static analysis attributes, tree-based ensemble approaches are quite successful in detecting malware.

The Isolation Forest model, on the other hand, fared far worse when tested as an unsupervised anomaly detection method. With an F1-score for malware below 0.13 and an overall accuracy of only 41.7%, it was unable to identify the dataset's discriminative patterns. This outcome is in line with predictions since supervised learning techniques are more appropriate when labeled malware and benign samples are available, while isolation forest is better suited for novelty identification in highly imbalanced or unlabeled datasets.

Copyright © 2025 by **I**nternational **J**ournal on **O**ptimization and **A**pplications

VI. CONCLUSION

All things considered, the tests show that supervised learning models (Random Forest, XGBoost) greatly outperform unsupervised methods like Isolation Forest in static malware detection based on hashing and fuzzy signatures. These results confirm how well machine learning and static analysis work together to provide reliable malware detection.

REFERENCES

[1] « Endpoint Detection and Response: Why Use Machine Learning?,https://doi.org/10.1109/ICTC46691.2019.8939836 ».

[2] « Valentine Machaka, and Titus Balan 2. Investigating Proactive Digital Forensics Leveraging,https://doi.org/10.3390/app12189077. 2022 ».

[3] « Fagbohunmi Grifin Siji, Okafor Patrick Uche. An improved model for comparing different endpoint detection and response tools for mitigating insider threat,https://doi.org/10.54905/disssi/v20i53/e22ije1651. 2023 ».

[4] « Patsakis, George Karantzas and Constantinos. https://doi.org/10.3390/jcp1030021. »

[5] « Nilam Nur Amir Sjarif, Suriayati Chuprat,Mohd Naz'ri Mahrin,Noor Azurati Ahmad,Aswami Ariffin,Firham M Senan,Nazri Ahmad Zamani,Afifah Saupi. Endpoint Detection and Response: Why Use Machine,https://iopscience.iop.org/article/10.1088/1742-6596/2062/1/012013. 2021. »

[6] « Markus Wurzenberger, Georg Höld, Max Landauer, Florian Skopik. Analysis of statistical properties of variables in log data for advanced anomaly detection in cyber security,https://doi.org/10.1016/j.cose.2023.103631. 2023. »

[7] « Breast Cancer Risk Prediction using XGBoost and Random Forest Algorithm,https://doi.org/10.1109/ICCCNT49239.2020.9225451 ».

[8] « An Improved Data Anomaly Detection Method Based on Isolation Forest ,https://doi.org/10.1109/ISCID.2017.202 ».

[9] « Fuzzy-Import Hashing: A Malware Analysis Approach,https://doi.org/10.1109/FUZZ48607.2020.9177636 ».

[10] « On Enhancing the Minimum Hamming Distance of Polar Codes,https://doi.org/10.1109/SPAWC.2016.7536756 ».

[11] « An overview of obfuscation techniques used by malware in Visual Basic for Application scripts,https://doi.org/10.1109/SYNASC.2018.00051 ».

[12] « Cognitive Modeling of Polymorphic Malware Using Fractal Based Semantic Characterization ,https://doi.org/10.1109/THS.2017.7943487 ».

[13] « NIST, Secure Hash Standard (SHS), FIPS PUB 180-4, 2015 ».

[14] « Kornblum, J. Identifying Almost Identical Files Using Context Triggered Piecewise Hashing (SSDEEP original paper) ».

[15] « R. Hamming, "Error Detecting and Error Correcting Codes, ».