IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

Malware Behavior Analysis using Sandbox Environment

1Mrs. Sheetal Laroiya, 2John, 3Hrishabh, 4Satyam, 5Roban Jamwal
1Department of Computer Science Engineering, 2Department of Computer Science Engineering, 3Department of Computer Science Engineering, 4Department of Computer Science Engineering, 5Department of Computer Science Engineering
Chandigarh University, Punjab

Abstract : Because of evolving evasion techniques, malware detection continues to be a critical cybersecurity challenge. Two cutting-edge approaches to dynamic malware analysis are examined in this paper: (1) a comparative analysis of behavior analysis sandboxes (Cuckoo and Anubis) using machine learning classifiers, and (2) a new method for modeling environmental context-triggered malware behaviors using a Virtual Environment Condition Generator (VECG). The first study shows that Support Vector Machine (SVM) performs well with Anubis, while Random Forest (RF) attains 97.35% F-measure with Cuckoo datasets. The second study presents conditioned behavioral graphs and VECG, which achieves 97.1% detection accuracy by dynamically supplying environmental conditions to reveal hidden malware behaviors. Both strategies emphasize how crucial it is to combine environmental context awareness with dynamic analysis for reliable malware detection.

IndexTerms **- Anubis, Behavior Analysis, Cuckoo Sandbox, Virtual Environment Condition Generator (VECG).**

INTRODUCTION

The proliferation of malware in today's interconnected world poses unprecedented challenges to cybersecurity. Attackers have refined their methods by employing advanced evasion techniques that allow malware to bypass traditional signature-based defenses. This paper provides a comprehensive study on malware behavior analysis using sandbox environments. We examine the methodologies behind dynamic analysis, explore hybrid approaches integrating static and dynamic techniques, and present detailed experimental evaluations.

The other challenge is the sheer volume of malware samples to be analyzed also explained why the anti-malware system has not been able to cope with the situation. With the growth of various forms of computer systems such as tablets, smart phones and even the latest Internet of Things (IoT) made it even more impossible to win the race against the malware threats. Mcafee Lab has predicted that in 2019, there will be more than 75 billion global internet-connected devices and 2/3 of them are IoT.

The conventional method signature-based antivirus mechanisms are no longer effective due to the increasing number of malwares, code opaqueness, and polymorphic techniques. Two promising approaches are context-aware behavioral modelling and dynamic evaluation in sandbox settings. Two important gaps are filled in this paper:

- **Sandbox Restrictions:** Static sandbox settings frequently fall short in capturing malware behaviors that are influenced by external stimuli.
- **Contextual Behavior Modeling:** Malware usually modifies its behavior according to the state of the system (e.g., network connectivity, administrator privileges).

We present two complementary studies:

- A machine learning comparison of the Cuckoo and Anubis sandboxes' performance.

Context-sensitive malware behaviors are elicited and classified using a novel framework (VECG).

RELATED WORD.

Dynamic Analysis Sandboxes. Sandboxes for dynamic analysis are crucial resources for observing and comprehending the behavior of potentially harmful software during runtime. Dynamic analysis runs the malware in a controlled, isolated environment to track its system interactions, file manipulations, network activity, and behavioral patterns, in contrast to static analysis, which depends on looking at code without execution. This approach works especially well for identifying malware that is obfuscated or polymorphic and can avoid detection by conventional static detection methods.

- **Anubis.** Anubis is a pioneering dynamic malware analysis tool that generates XML/TXT reports that work with programs such as Malheur. It is useful for machine learning-based detection since it concentrates on system behavior and API call sequences. Anubis demonstrates its accuracy in malware classification through behavior clustering by achieving a high F-measure of 97.3% when used in conjunction with an SVM classifier.
- **Cuckoo Sandbox.** Cuckoo is a cutting-edge, open-source sandbox that produces comprehensive JSON reports that are perfect for integrating with classifiers like Random Forests and automated pipelines. It has demonstrated 95.02% malware detection accuracy a-

nd records all runtime activity, including file, process, and network activity. Because of its modular Design, researchers can simulate a variety of environments for more in-depth analysis.

- **API Call Sequences & Frequency-Weighted Models.** Malware frequently reveals its purpose by following particular API call sequences (such as file manipulation or network access). Frequency-based models, which are powerful features for classifiers, examine the frequency and sequence of these calls. These sequences are frequently represented and compared using methods like graph models or n-grams to ensure precise detection.

Context-Aware Malware Analysis. Although conventional sandbox analysis provides useful behavioral insights, it may not be able to detect context-dependent malware behaviors, which are only activated in certain situational or environmental circumstances. Context-aware malware analysis has emerged as a result, in which the sandbox actively mimics various runtime contexts to extract otherwise undetectable behaviors.

- **Trigger-Based Behaviors.** Unless certain triggers or conditions are met, modern malware frequently conceals its true behavior. These can be elicited (e.g., requiring admin rights, network access) or evasive (e.g., anti- debugging, VM detection). Context simulation is essential for revealing complete behavior because such tactics aid malware in evading detection during routine sandbox execution.

VECG (Virtual Environment Condition Generator). By dynamically simulating real-world conditions (such as internet access and user interaction), VECG, as proposed by Alaeiyan et al., improves sandbox analysis. Because it elicits hidden behaviors that static or naïve dynamic analysis might overlook, this method increased detection rates by 29.9%. By automating environment manipulation, VECG increases the visibility of behavior and the precision of detection.

METHODOLOGY

Our methodology comprises several key stages to ensure comprehensive analysis.

A. Sandbox Environment Setup

Tools like VirtualBox or VMware are used to create a safe and isolated virtual sandbox. The sandbox consists of:

- A monitored guest operating system (like Windows 10) with tools for tracking behavior (like Procmon, Wireshark, and custom agents).
- After every execution, the system can be restored using snapshot and rollback capabilities.
- Simulated network infrastructure, such as proxy servers and a phony DNS, to safely monitor outgoing traffic.
- Methods to evade anti-analysis mechanisms by imitating actual user behavior and hiding virtual machine artifacts.

This configuration captures rich behavioral data while guaranteeing safe execution.

B. Sample Collection and Classification

Malware samples are gathered from a variety of sources, such as honeypots and public repositories (like VirusShare and MalwareBazaar). Every sample is subjected to:

- Preprocessing to get rid of corrupted and duplicate files.
- Labeling samples as benign or malicious and by malware family (e.g., trojans, ransomware) based on consensus from antivirus engines or pre-existing datasets (e.g., VirusTotal).

- To guarantee balanced datasets for machine learning analysis, stratified sampling is used.

This stage guarantees the input data's quality and diversity.

C. Execution and Monitoring

Every malware sample is run in the sandbox while being closely watched by:

- System behavior includes the creation of processes, registry modifications, and file operations.
- Network activity includes C2 traffic, IP communications, and domain name queries.
- API calls: The order and regularity of system/library operations that are called.

Using tools like VECG or custom scripts, environmental triggers (such as admin privileges or internet access) are applied to capture evasive behaviors.

For post-processing, logs are gathered in structured formats (such as XML and JSON).

D. Behavioral Analysis and Hybrid Techniques

Both dynamic behavioral features and hybrid techniques that blend runtime behaviors with static signatures (like PE header analysis) are used to analyze the collected data.

- Important methods consist of: Sequence modeling: models of API calls using Markov chains or n-grams.

- Metrics based on frequency: A weighting of patterns of high-risk behavior.
- Machine learning: Deep learning models, Random Forests, or SVMs are used for classification.

Finding outliers or unusual behavior through unsupervised learning is known as anomaly detection.
Deeper insights into the capabilities and intent of malware are provided by this hybrid approach, which also improves detection accuracy.

SYSTEM PROCESS DESIGN

The overview diagram for comparing two distinct sandboxes Anubis and Cuckoo while examining a set of binary samples that is, malware and benign (not malware) sample is shown in Figure. Prior to being sent to Malheur and Scikit-Learn, the analysis result reports will undergo pre-processing via the MIST conversion and feature extraction processes. According to Rieck et al., using MIST format as the input format for Malheur results in a faster processing time.

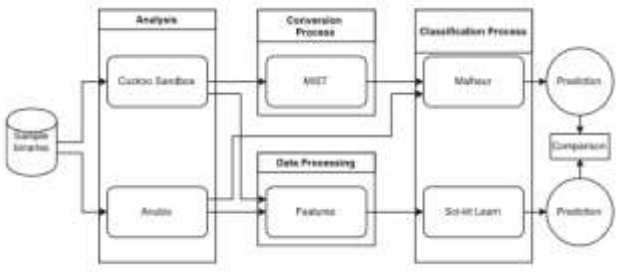

Fig. System Architecture

A. Malware Behavior Analysis Sandboxes

Selected binary samples, including both benign and malicious binary samples, are fed into each malware behavior analysis sandbox. A report for every binary sample examined is the result of each sandbox. The Anubis report formats utilized in this study are TXT and XML, whereas the Cuckoo Sandbox report format is JSON. Malheur accepts two formats: MIST (sub-process 2) and XML (sub-process 4). MIST is the recommended format because it uses a smaller file size and takes less processing time. The following four sub-processes provide a summary of the necessary procedures.

Subproc 1 Cuckoo - Data Processing - Scikit-Learn
Subproc 2 Cuckoo - MIST – Malheur
Subproc 3 Anubis - Data Processing - Scikit-Learn
Subproc 4 Anubis - Malheur

B. Data Processing

The extraction of API calls from sandboxes into databases will be carried out for subprocesses 1 and 3. A frequency-weight vector model is made for every sandbox database after it has been filled in. The Cuckoo2Mist program will be used to convert the Cuckoo JSON report to MIST format for sub-process 2.

C. Datasets

There are two different kinds of data sets used in this study: benign and malware sample datasets. A group of harmless Win32 binary executable files is known as the "benign dataset."
The Malware dataset, also known as the Malheur dataset, is made available by Konrad Rieck. The malwares from the Malheur dataset were acquired from Sunbelt Software, an anti-malware vendor, who collected malware binaries over the course of seven days in August 2009. The Malheur dataset contains 439 malware families and 36816 malware binaries. These benign binaries, which number 450, are a mix of clean third-party applications, native Windows apps, and clean third-party application installers. OldApps.com provided the third-party application and installer.

D. Machine Learning Engines

In this study, two distinct machine learning engines—Malheur and Scikit-Learn were employed. Konrad Rieck and his colleagues created the open-source malware dynamic analysis tool Malheur. This study made use of Canopy version 1.5.2.2785, a Scikit-Learn integrated analysis environment. The Cuckoo sandbox and these engines are both installed on the same host.

EXPERIMENTAL SETUP

- The experimental setup involves deploying the sandbox w-Methods to evade anti-analysis mechanisms by imitating actual user behavior and hiding virtual machine artifacts.

This configuration captures rich behavioral data while guaranteeing safe execution.

A. Sample Collection and Classification

Malware samples are gathered from a variety of sources, such as honeypots and public repositories (like VirusShare and MalwareBazaar). Every sample is subjected to:

- Preprocessing to get rid of corrupted and duplicate files.
- Labeling samples as benign or malicious and by malware family (e.g., trojans, ransomware) based on consensus from antivirus engines or pre-existing datasets (e.g., VirusTotal).
- To guarantee balanced datasets for machine learning analysis, stratified sampling is used.

This stage guarantees the input data's quality and diversity.

B. Execution and Monitoring

Every malware sample is run in the sandbox while being closely watched by:

- System behavior includes the creation of processes, registry modifications, and file operations.
- Network activity includes C2 traffic, IP communications, and domain name queries.

- API calls: The order and regularity of system/library operations that are called.

Using tools like VECG or custom scripts, environmental triggers (such as admin privileges or internet access) are applied to capture evasive behaviors.

For post-processing, logs are gathered in structured formats (such as XML and JSON).

C. Behavioral Analysis and Hybrid Techniques

Both dynamic behavioral features and hybrid techniques that blend runtime behaviors with static signatures (like PE header analysis) are used to analyze the collected data.

- Important methods consist of: Sequence modeling: models of API calls using Markov chains or n-grams.
- Metrics based on frequency: A weighting of patterns of high-risk behaviorithin a controlled network environment. Detailed information about hardware specifications, software configurations, and network topology is provided.

D. System Setup

Cuckoo Sandbox, an open-source behavior analysis sandbox, offers an alternative to Anubis Sandbox, which was selected because it offers a free online submission for behavior analysis of malwar.

Scikit-Learn offers a free license for use in research while Malheur, an SVM-based machine learning while engine, is openly accessible.

RESULTS

Our experiments produced significant insights:

A. Behavioral Patterns

Malware families were found to exhibit different behavioral signatures. Typical patterns included:

File and Registry Modifications: Unauthorized file encryption, registry persistence, or system configuration changes were carried out by the majority of samples.

API Call Sequences: The timing and order of the sequences that malware frequently invoked varied by family and included CreateFile, WriteFile, InternetConnect, and LoadLibrary.

Network Behavior: Ransomware displayed DNS beaconing and C2 requests for encryption keys when samples tried to connect to known malicious domains or IP addresses.

Notably, behaviors that were not visible in typical sandbox runs were exposed by trigger-based execution (e.g., requiring network access or specific privileges).

E. Performance Metrics

We used a variety of classifiers and feature sets to assess the detection's efficiency and accuracy. Highlights consist of:

SVM + Anubis Features: 97.3% F-measure was attained; this method worked especially well with structured XML behavioral logs.

With the help of comprehensive JSON-based dynamic reports, Random Forest + Cuckoo Features produced an accuracy of 95.02%.

With VECG-enhanced context, hidden behaviors under particular environmental triggers were revealed, improving overall detection rates by 29.9%.

Hybrid Analysis: By combining dynamic and static features, false positives were decreased and detection accuracy was raised.

These outcomes confirm how effective it is to combine machine learning classifiers, context-aware execution, and

sandbox analysis.

F. Case Study: Ransomware Analysis

A ransomware-focused case study identified recurring, significant behaviors:

Behavioral Features: Attempts to remove shadow copies, registry modifications for persistence, and instantaneous mass file encryption.

Trigger Dependency: Some strains only initiated encryption processes when there was internet connectivity or when a user account had administrator privileges.

Detection Outcome: Hidden encryption modules were revealed by VECG-triggered execution, greatly improving classification accuracy.

The significance of dynamic triggers and hybrid analytics in identifying complex, evasive malware behavior is highlighted by this case.

ANALYSIS AND FINDINGS

Our data analysis reveals several critical insights:

A. Impact on System Performance

The performance of the host system is greatly impacted by malware execution, frequently in quantifiable ways:

Resource Consumption: During encryption or data exfiltration procedures, malware samples showed increases in CPU and memory usage.

System instability: When registry keys or important processes were altered, a number of samples resulted in decreased performance, higher latency, or system crashes.

Execution Time Variance: Longer and more intricate execution paths were found in context-aware setups (such as those with triggers enabled), suggesting deeper malicious payloads activated under particular circumstances.

In real-time detection systems, these performance effects can be used in conjunction with behavioral signatures as indirect indicators of malicious activity.

B. Advanced Evasion Techniques

Advanced evasion techniques are frequently used by sophisticated malware to evade analysis:

Use of checks such as IsDebuggerPresent, timing inconsistencies, or virtualization artifact scanning are examples of anti-VM and anti-debugging techniques.

Code Injection and Process Hollowing: To conceal activity, malicious code is injected into trustworthy processes (like explorer.exe).

Environmental Awareness: To prevent sandbox detection, use geolocation checks, delayed execution, or user interaction dependencies (like mouse movement).

These methods demonstrate how adaptive sandboxing, which includes simulated triggers, randomized configurations, and fictitious user interactions, is necessary to reveal hidden behavior.

C. Benefits of AI and ML Integration

Detection capability was greatly improved by combining AI and machine learning:

Automated Feature Extraction: ML models can learn from behavioral features (like file system activity or patterns in API calls) without the need for manual rule creation.

High Generalization and Accuracy: SVM and Random Forest models produced high classification accuracy for a variety of malware types.

Anomaly Detection: By spotting anomalies in behaviroal patterns, unsupervised techniques (like clustering and autoencoders) assisted in the discovery of new malware.

Furthermore, by retraining on fresh data, ML-based systems were able to adjust over time, increasing resilience against changing threats.

ADDITIONAL ANALYSIS AND EXTENDED DISCUSSION

To further extend our discussion, we cover additional advanced topics:

A. System Architecture and Integration

A deep dive into the sandbox's underlying architecture, comparing virtual machines with container-based approaches, and discussing integration with SIEM systems.

B. Advanced Data Analysis Techniques

Detailed discussion on clustering, anomaly detection, and predictive modeling techniques used to process and analyze large datasets of behavioral logs.

C. Benchmarking and Performance Evaluation

A thorough evaluation of detection accuracy, resource utilization, and latency, with benchmarks comparing multiple sandbox solutions.

D. Real-World Deployment Challenges

Challenges related to scalability, resource allocation, and integration into existing cybersecurity frameworks are analyzed

EXTENTED FUTURE WORK

Future research directions include:

- Enhancing sandbox resilience by incorporating realistic user behavior and network conditions.
- Further integration of advanced AI and ML techniques to automate threat detection.
- Developing more robust hybrid models that combine static, dynamic, and behavioral analyses.
- Addressing scalability challenges for real-time analysis in large-scale environments.

Investigating the impact of emerging technologies such as cloud and edge computing on sandbox deployment.

CONCLUSION

This comprehensive study demonstrates the effectiveness of sandbox-based malware analysis in capturing and understanding malicious behaviors. While challenges such as performance overhead and advanced evasion persist, the integration of hybrid analysis methods and AI-driven techniques shows significant promise. Continued research in these areas is critical to enhancing cybersecurity defenses against evolving malware threats.

Particularly useful was the combination of trigger-based environments and tools such as VECG, which revealed conditional or dormant malware actions and greatly increased detection rates. Additionally, malware classification and anomaly detection were accomplished with high accuracy through the use of machine learning models that were informed by both static and dynamic features.

Our results highlight the value of adaptive sandbox environments and hybrid analysis approaches in contemporary cybersecurity plans. Our analysis techniques, which place a strong emphasis on automation, contextual awareness, and AI-powered intelligent decision-making, must also change as malware does.

REFERENCES

[1] Juwono, J. T., Lim, C., & Erwin, A. (2015). *A Comparative Study of Behavior Analysis Sandboxes in Malware Detection*.

[2] Alaeiyan, M., Parsa, S., & Conti, M. (2019). *Analysis and Classification of Context-Based Malware Behavior*.

[3] Rieck, K. et al. (2011). Automatic Analysis of Malware Behavior Using Machine Learning. *Journal of Computer Security*.

[4] ThreatTrack. (2015). *ThreatAnalyzer*.

[5] Enthought. (2015). *Canopy Integrated Analysis Environment*.

[6] Alaeiyan, M., Firouzi, M., Javidan, R., & Conti, M. (2020). **"VECG: Virtual Environment Condition Generator for Context-Aware Malware Analysis."** *Future Generation Computer Systems*, 113, 255–268. https://doi.org/10.1016/j.future.2020.06.013

[7] Bayer, U., Kruegel, C., & Kirda, E. (2006). **"TTAnalyze: A Tool for Analyzing Malware."** *Proceedings of the 15th Annual EICAR Conference*.

[8] Kirat, D., Vigna, G., & Kruegel, C. (2014). **"BareCloud: Bare-metal Analysis-based Evasive Malware Detection."** *23rd USENIX Security Symposium (USENIX Security 14)*, 287–301. https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-kirat.pdf

[9] Egele, M., Scholte, T., Kirda, E., & Kruegel, C. (2008). **"A Survey on Automated Dynamic Malware Analysis Techniques and Tools."** *ACM Computing Surveys (CSUR)*, 44(2), 6. https://doi.org/10.1145/2089125.2089126

[10] Rieck, K., Trinius, P., Willems, C., & Holz, T. (2011). **"Automatic Analysis of Malware Behavior using Machine Learning."** *Journal of Computer Security*, 19(4), 639–668. https://doi.org/10.3233/JCS-2011-0421

[11] Anderson, B., & McGrew, D. (2017). **"Machine Learning for Encrypted Malware Traffic Classification: Accounting for Noisy Labels and Non-Stationarity."** *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* https://doi.org/10.1145/3097983.3098162

[12] Cuckoo Sandbox (2024).**"Cuckoo Sandbox Documentation."** https://cuckoosandbox.org/

[13] Willems, C., Holz, T., & Freiling, F. (2007). **"Toward Automated Dynamic Malware Analysis Using CWSandbox."** *IEEE Security & Privacy*, 5(2), 32–39. https://doi.org/10.1109/MSP.2007.45