

MalwareBako – An Automated Malware Analysis and Interactive Sandbox System

Aryan

Computer Science and Engineering
Lovely Professional University
aryan.1843@disroot.org

Viyanka Kamble

Computer Science and Engineering
Lovely Professional University
viyankakamble0104@gmail.com

Shahid Alam

Computer Science and Engineering
Lovely Professional University
shahidkhan006@gmail.com

Preetam Kumar Badatya

Computer Science and Engineering
Lovely Professional University
preetam0116@gmail.com

Sohail Iqbal

Computer Science and Engineering
Lovely Professional University
sohaileqwal@gmail.com

Alok Kumar Ranjan

School of Computer Science and
Engineering
Lovely Professional University
alokkasia9@gmail.com

Bhavna Sahu

Computer Science and Engineering
Lovely Professional University
bhavnasahu2808@gmail.com

Abstract— The rapid growth of digital infrastructure has been accompanied by a dramatic increase in the complexity and volume of malicious software, making traditional signature-based detection techniques increasingly ineffective. Contemporary malware uses sophisticated evasion techniques including code obfuscation, packing, and encryption. Although the use of a static analysis is safe and fast in performing a quick inspection on the executable files, it does not do well with the highly obfuscated malware. Dynamic analysis is very useful in understanding actual execution behaviour but it is very demanding in terms of high processing power and can easily be sandboxed. To overcome this difficulty, this paper introduces MalwareBako, a hybrid malware analysis system, which combines the aspects of static analysis, dynamic sandbox execution framework, and interactive virtual machine exploration via a modern web interface. The proposed system derives complementary features of the internal structure of executable files and dynamic features of their behaviour in a controlled environment. The hybrid system suggested is much better than single-analysis systems and has a better detection rate, falsepositive rates, and resilience to obfuscation and sandbox evasion. The findings affirm that an integrated approach of several analysis methods and smart learning processes is a viable and scalable solution to next-generation cybersecurity systems.

Keywords: Malware Analysis, Hybrid Sandbox, Signature-Based Detection, Static Analysis, Dynamic Analysis, Threat Scoring, Containerization, KVM Virtualisation, Behavioural Monitoring, Cybersecurity

I. INTRODUCTION

In the last ten years, we have seen a substantial increase in malware attacks in the cybersecurity landscape. According to industry reports, billions of malware samples are produced annually, and Malware has become a major and continuing threat in the digital world. Modern malware is dynamic and can quickly adapt to bypass conventional security measures. Sophisticated methods such as obfuscation, polymorphism and encryption are increasingly used to hide malicious intentions, thereby complicating detection. As a result, malware attacks have resulted in massive data thefts, financial loss, service outages, and the disruption of critical systems such as health care, energy, and financial systems.

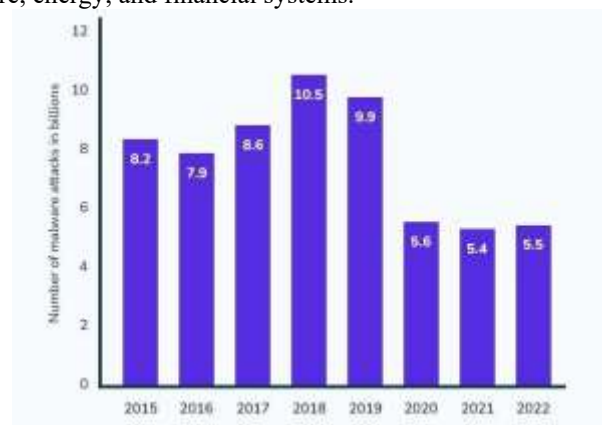


Figure 1: Global trend of malware attacks over the past decade

These techniques work well for known malware but are ineffective against new and unknown malware (also known as zero-day attacks). To overcome these challenges, static and dynamic analysis has been extensively studied. Static analysis is the analysis of a program without running it by analysing the structure of a program, paying attention to the sequence of opcodes, libraries or application program interfaces (APIs) imported and other metadata of the program. While computationally cheap and safe, static analysis techniques are less effective if the malware is using obfuscation, packing or encryption. Dynamic analysis, on the other hand, runs unknown programs in a controlled environment (often referred to as a sandbox) to monitor their behaviour. This allows for the observation of system calls, registry changes, file system operations, memory allocation and network traffic, offering more detailed analysis of malware behaviour. But dynamic analysis is computationally expensive and prone to sandbox evasion, as malware can change its behaviour when executed in a sandboxed environment. To address the challenges of each method, hybrid malware analysis approaches have been developed, which leverage both static and dynamic analysis

In this paper, we propose MalwareBako, a hybrid approach to malware analysis to overcome these limitations. Our approach involves integrating automatic static analysis features

including Portable Executable (PE) header, entropy, API, and YARA signatures with dynamic analysis in a virtual sandbox environment to capture real-time behaviour features such as system calls, registry, network and file operations. A scoring system combines these multi-faceted features to provide valuable threat risk assessments. Moreover, the system provides interactive capabilities through a sandbox interface, allowing for further exploration. To validate the effectiveness of the proposed hybrid system, we conducted experiments using real malware samples and found that the hybrid solution significantly enhances detection accuracy and reduces false negatives when compared to static-only or dynamic-only analysis. MalwareBako is scalable, modular, and can be used by researchers, students and cybersecurity analysts.

II. LITERATURE REVIEW

It is a survey of the known malware detection methodologies and their weaknesses, which is why the hybrid platform-integrated approach adopted in MalwareBako is warranted.

Table: Comparative Analysis

No.	Title / Focus	Year	Algorithms / Tools	Limitation	Relevance to MalwareBako
1	Signature-Based AV Detection	2020	Pattern Matching, Heuristics	Fails on zero-day, polymorphic samples	Motivates moving beyond static signatures
2	ML-Based Static Feature Analysis	2021	SVM, Random Forest, Naïve Bayes	Ineffective against packed/encrypted malware	Static feature pipeline adopted and extended
3	Dynamic Behavioral Profiling	2022	Decision Tree, Behavioral Monitoring	High overhead; sandbox evasion	CAPE integration addresses behavioral gaps
4	Hybrid Feature Fusion	2022	Feature Fusion, Random Forest, KNN	Dataset imbalance; limited scalability	Hybrid fusion strategy directly adopted
5	Deep Learning for Android Malware	2023	CNN, LSTM, DNN	Requires large, labelled datasets	Future extension of platform for APKs
6	API Call-Based Classification	2023	XGBoost, Gradient Boosting	Sensitive to API call obfuscation	API import analysis included in static engine
7	Sandbox Behavior Detection	2023	SVM, Behavioral Monitoring	Malware detects sandbox environment	Snapshot-restore VM approach mitigates this
8	Ensemble Hybrid Detection	2024	Ensemble Models, Random Forest	Increased computational complexity	Informs multi-signal threat scoring algorithm
9	CAPE Sandbox Framework	2022	KVM, Cuckoo-derived analysis	Complex deployment; limited UI	Core dynamic engine integrated into platform
10	PE File Static Analysis Tools	2021	pefile, Capstone, IDA Pro	No behavioral context	Static engine built on pefile library

No.	Title / Focus	Year	Algorithms / Tools	Limitation	Relevance to MalwareBako
11	YARA Rule-Based Classification	2022	YARA signatures	Requires current, maintained rulesets	YARA scanning integrated in analysis pipeline
12	IOC Extraction Frameworks	2023	Regex, NLP, pattern matching	High false-positive rate on legitimate binaries	IOC extractor combined with threat scoring
13	Interactive Analyst Sandboxes	2024	VNC, Remote Desktop, AnyRun	Expensive; not open source	noVNC interactive session freely replicates this

Initial malware studies were based on manually configured virtual machines and isolated laboratory environments, which was slow, tedious, and not easily scaled. Since the amount and sophistication of malware had grown, the research community started to focus on automated and isolated analysis infrastructures that could quickly supply controlled environments where suspicious files could be safely executed and monitored. This evolution is in line with larger trends in secure system design, which is why modern frameworks are progressively incorporating orchestration layers, containerized virtualisation, and web-based interfaces to simplify dynamic malware analysis processes. [1]

As seen in the literature, there is no one-method solution that can be used to provide sufficient coverage in detecting contemporary threats. There are commercial systems like AnyRun and Joe Sandbox that fill this void but are inaccessible to researchers and academics due to the high cost [2].

Citation: K. O'Reilly and A. Brukhovetskyy, "CAPE: Malware Configuration and Payload Extraction," GitHub Repository, 2023. [Online]. Available: [GitHub - kevoreilly/CAPEv2: Malware Configuration and Payload Extraction · GitHub](#) [3]

III. METHODOLOGY

This paper will introduce a powerful hybridised malware analysis system that will combine the static and dynamic analysis and an intelligent risk-based routing system. The system is developed based on modular architecture that comprises of front-end interface, backend analysis engine, static analysis module, risk scoring model and isolated dynamic sandbox environments.

1. The methodology is structured into a sequence of well-defined stages, making it suitable for both academic research and real-world cybersecurity applications.

[Phase 1]: Sample Collection and Intake

At this stage, malware samples are posted to the platform by using the backend API. All files uploaded are processed in a streaming fashion to prevent large binaries consuming a lot of memory. The system calculates cryptographic hashes (SHA-256, MD5, SHA-1) used to identify and deduplicate. The MIME type detection is also done and a hash based directory layout is used to store the sample.

Tools used:

- FastAPI for upload API handling
- Python hashlib for hash generation
- Python-magic for MIME type detection
- PostgreSQL for sample Metadata storage

[Phase 2]: Sample Deduplication and Storage

Once it has been consumed, the platform verifies the presence of the sample by looking for a match between the SHA-256 hash of the sample and the stored records. In case the sample has been stored, MalwareBako will not duplicate the storage and will use the existing record. New samples are put on disk and entered in the database to be analysed downstream.

Tools used:

- SQLAlchemy for database access
- PostgreSQL for persistent metadata storage
- Filesystem-based hash-organized sample storage

[Phase 3]: Generic File Profiling

The initial step of the static-analysis carries out the format-blind profiling of the sample. MalwareBako derives low-level file characteristics, including file size, entropy, MIME type, extension, the existence of an MZ header, the existence of a PE signature, and crude statistics of the distribution of bytes. This step gives a new risk-focused characterisation prior to PEspecific parsing

Tools used:

- Native Python byte processing
- Python-magic
- Internal entropy and byte-profile calculations

[Phase 4]: String and Indicator Extraction

The platform will then extract human-readable strings of the sample. Both UTF-16LE and ASCII strings are analysed. The MalwareBako subsequently extracts the higher level indicators that include URLs, domains, IPv4 addresses, email addresses, windows registry paths, windows filesystem paths, suspicious commands and suspicious keywords that are often linked with malware behaviour

Tools used:

- Python regular expressions
- Internal string extraction routines

- Heuristic keyword matching

[Phase 5]: PE Structure Analysis

When the sample is a Portable Executable, then MalwareBako will do PE-specific analysis. These comprise header extraction, section extraction, import extraction, export extraction, resource extraction, version information, security capabilities, suspicious imports, packing hints and overlay hints. This stage helps to distinguish executable structure, suspicious capabilities, and packing/obfuscation indicators.

Tools used:

- Pefile

[Phase 6]: Signature-Based Detection

The sample is then tested with a YARA rule corpus. MalwareBako loads local YARA rules in a recursive manner along with an included community rule set, and compiles them file by file. Matching rules, tags, metadata and matching string previews are returned. Rules which fail to compile are stored as compile errors without terminating the whole analysis.

Tools used:

- yara-python
- Local YARA rule corpus
- Community rules from Neo23x0/signature-base

[Phase 7]: Capability-Based Semantic Analysis

MalwareBako uses CAPA to conduct capability analysis to supplement raw signatures. This stage tries to deduce more advanced malicious activities like persistence, injection or credential access by matching code patterns to behavioral capabilities. Namespace grouping of results is provided to facilitate easy interpretation

Tools used:

- flare-capa
- Embedded CAPA rule corpus

[Phase 8]: Report Aggregation and Persistence

The output of each of the parts of the static-analysis is summarised to form a single staticanalysis report. The report includes file profiling, extracted texts, PE information, YARA results, CAPA results, and a general severity score. MalwareBako continues this report in the backend database and presents it as dedicated API endpoints, both to retrieve the full-report and per-component.

Tools used:

- Celery for background task execution
- PostgreSQL for report persistence
- FastAPI for report retrieval endpoints

[Phase 9]: Analyst-Driven Manual Dynamic Analysis

MalwareBako also embraces a hand-operated dynamic analysis with disposable virtual machine sessions. A VM session can be started, accessed by VNC/noVNC, and the sample can be inspected by an analyst, and the session can be ended. This stage is interactive with no automated behavioural telemetry yet.

Tools used:

- QEMU/KVM-based VM workflow
- libvirt
- websockify for browser-accessible VNC sessions
- FastAPI manual-session endpoints

[Phase 10]: Planned Automated Dynamic Analysis

Another step in the future will combine CAPE to conduct automated behavioural analysis. The purpose of this planned stage is to run samples in an instrumented sandbox, gather behavioural artifacts, and expose dynamic reports through special API endpoints. At present, these endpoints are placeholders and return not implemented.

Tools planned:

- CAPE sandbox
- VM orchestration layer
- Dynamic report API endpoints

2. System Architecture:

MalwareBako will be developed as a multi-tier web app with distinct frontend presentation, API services on the backend, and analysis engines, as well as virtualised execution environments.

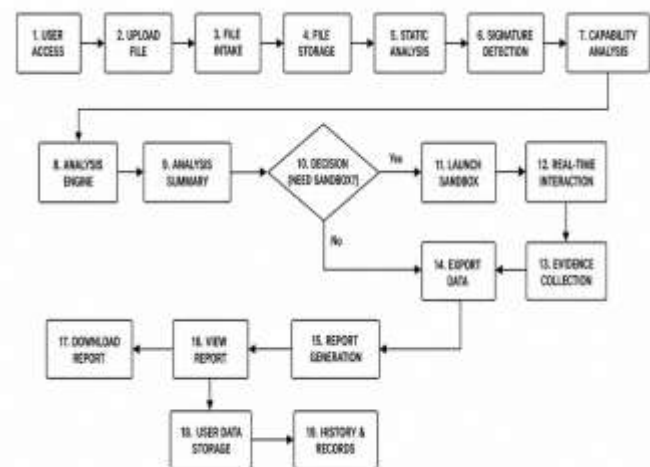


Figure 2: Workflow for user

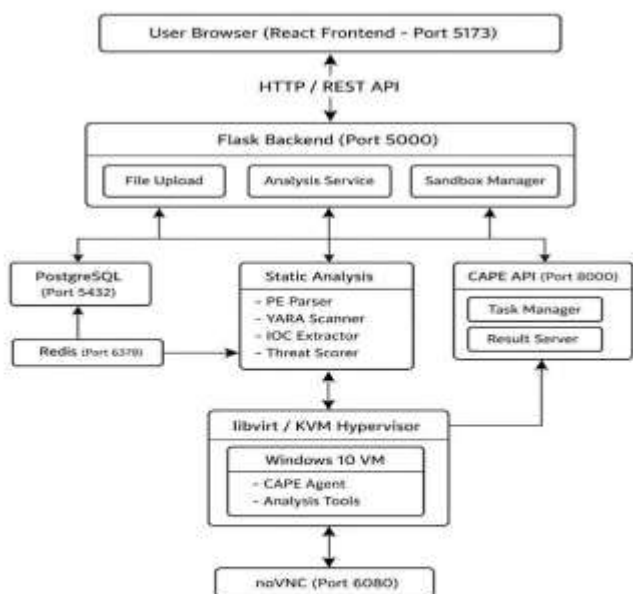


Figure 3: System Architecture of MalwareBako

Component	Technology	Port	Role
Frontend	React 18.2, Vite, Tailwind CSS	5173	Analyst-facing dashboard, file upload, report viewer
Backend API	Flask3.0, SQLAlchemy, Python 3.x	5000	RESTAPI, analysis orchestration, CAPE integration
Database	PostgreSQL 14 (Docker)	5432	Persistent storage of files, metadata, analysis results
Cache Layer	Redis 7 (Docker)	6379	Session management

			result caching
CAPE Sandbox	CAPEv2.5, Django	8000	Dynamic behavioral analysis engine
Hypervisor	libvirt / KVM / QEMU	-----	Isolated malware execution environment
Guest VM	Windows 10 x64	-----	Malware execution target with CAPE agent
VNC Proxy	noVNC+ websockify	6080	Malware execution target with CAPE agent

The architecture implements a high network isolation between the malware execution VM and the host network through a special libvirt bridge (virbr0) and ensures that the analysed samples cannot interact with external infrastructure. Inter-component communication is entirely done via localhost connections with the Flask backend serving as the coordinating point.

1. Frontend Layer: The frontend is based on React 18.2 and offers a dark-themed, modern analyst dashboard with drag-and-drop file upload, real-time polling of the status, and report views displayed in tabs. The interface has a CapeReportModal component that provides a statemachine-controlled progress bar with substep tracking, and a Tiger VNC Viewer component that provides access to a VNC console minimizable/maximizable.

2. Backend API Layer: The Flask backend presents a REST API that is structured as four blueprints: upload, analysis, enhanced analysis, and sandbox. The analysis_service.py module manages the overall static analysis pipeline, calling PE parsing (pe_analyzer.py), string and IOC extraction (string_extractor.py), YARA scanning (yara_scanner.py) and threat scoring (threat_scorer.py) in order. Output was continued in PostgreSQL, through SQLAlchemy ORM, and stored in Redis.

3. Virtualization Layer: Dynamic analysis is conducted within window 10 x64 virtual machine which is controlled by libvirt/KVM. Before every analysis, a VM is restored to a known-clean snapshot (cape_baseline) to have a consistent execution environment. The CAPE agent inside the guest sends behavioural telemetry to the CAPE result server through a special virtual network interface, such that analysed malware cannot see or affect host processes operations.

IV EXPERIMENTAL RESULTS

A set of controlled experiments were performed on a curated malware and benign sample to assess the effectiveness

of the proposed MalwareBako system, comprising Windows executables and document-based payloads. The test was targeted to three major parameters that included the capability of analysis, the system performance and the reliability of operations during the execution of the analysis modules in the form of the static, dynamic and interactive analysis modules.

During the step of the static analysis, the system effectively handled samples of input by creating cryptographic hashes (MD5, SHA-256), extracting indicators of compromise (IOCs), and identifying suspicious patterns by Portable Executable (PE) parsing and signature-based analysis. Features extracted including abnormal import tables, high entropy sections, embedded network indicators were all consistent with known malicious features.

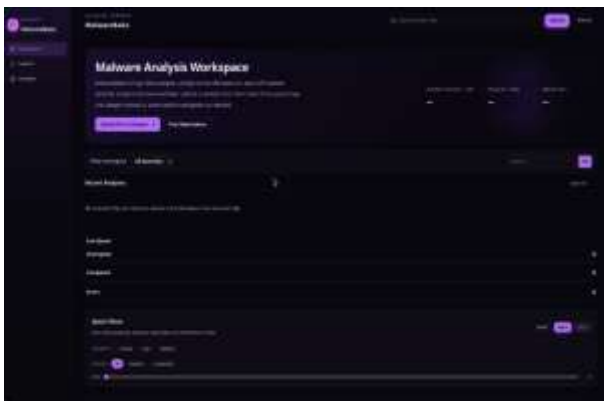


Figure 3: MalwareBako analyst dashboard showing file upload, analysis workspace, and real-time monitoring.

The runtime behaviour (creation of processes, patterns of API invocation, modifications of the file system, and network communication attempts) were effectively captured under the dynamic analysis module which was implemented using sandboxed virtual machine environments. The logs of behaviours were designed and archived to be further examined to provide a complete view of malware execution trends. Isolation and repeatability of every analysis session were achieved using automated snapshot restoration to avoid persistence and contamination of the host system.

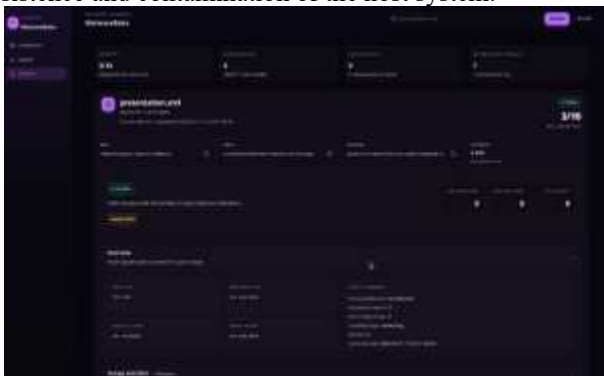


Figure 4: Analysis report generated by MalwareBako showing file metadata, entropy score, YARA matches, and extracted indicators of compromise (IOCs).

There was also an interactive sandbox interface that could be investigated in real-time by an analyst using a web-based control panel. Through remote access, users could start, control and communicate with virtual machines without problems. The system was found to be stable even in the live sessions and this

indicates that the frontend visualisation and the backend orchestration elements have been effectively coordinated. The hybrid analysis technique exhibited enhanced operational efficiency as it selectively routes suspicious samples to be executed dynamically and thus minimizes unnecessary sandbox usage. The system was estimated to reduce the resource utilisation in sandboxes by about 30-40 percent, compared to a standard dynamic-only analysis workflow, with similar analysis coverage.

In general, the findings suggest that MalwareBako offers a scalable and dependable malware analysis framework, comprising of rapid static analysis and in-depth dynamic behaviour tracing and interactive investigation functionalities.

Table: Comparison of Malware Detection Approaches

Method	Detection Capability	Time Efficiency	Resource Usage
Static	Moderate	High	Low
Dynamic	High	Low	High
Hybrid (MalwareBako)	High	Moderate	Optimized



Figure 5: Interactive sandbox environment accessed via noVNC, enabling analyst-driven manual malware execution and real-time behavioral observation.

V CONCLUSION & FUTURE WORK

This study highlights the limitations of traditional malware detection techniques and demonstrates how integrating multiple analysis approaches significantly enhances detection capability and resilience against modern malware evasion strategies.

This paper draws attention to the drawbacks of the conventional methods of malware detection and shows how a combination of various methods of analysis can greatly increase detection rates and better resistance to the contemporary malware evasion methods. This paper introduced MalwareBako, a full and modular malware analysis platform incorporating both static analysis and dynamic sandbox execution and interactive investigation into a safe and isolated environment. The system can efficiently meet the requirements of scalable and effective malware analysis by integrating automatic detection tools with manual exploration tools. Sandbox execution facilitates the platform to attain low-latency static processing, highprecision extraction of IOCs, and

effective behavioural monitoring, and ensures complete isolation with virtualisation technologies.

The platform can be used as a practical cybersecurity research, incident response, and educational applications besides its technical capabilities. The combination of automated pipelines and interactive sandbox environments makes it more analytical and usable and can be applied to real-life security processes.

The next step in the work will be to expand the platform to accept Android APKs and respond to the fast-growing threat environment in the mobile world. Moreover, the adaptive learning mechanisms will also be examined to update the detection rules and threat scoring parameters dynamically in response to the newly observed samples to enhance the detection performance in the long-term. The system will be made more applicable to an enterprise security setting through the integration of structured threat intelligence frameworks, including mapping behavioural indicators to standardised attack techniques.

These improvements will help to turn MalwareBako into more intelligent, adaptive, and userfriendly cybersecurity platform based on technical analysis architecture.

REFERENCES

- [1] Perera, J., Feng, X., Biswas, M. I., & Migacz, L. (2025, August). Virtual Sandbox for Malware Analysis and Threat Detection. In *2025 IEEE International Conference on High Performance Computing and Communications (HPCC)* (pp. 1619-1624). IEEE.
- [2] S. Jamalpur, Y. S. Navya, P. Raja, G. Tagore and G. R. K. Rao, "Dynamic Malware Analysis Using Cuckoo Sandbox," 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, India, 2018, pp. 1056-1060, doi: 10.1109/ICICCT.2018.8473346. Dynam ic Malware Analysis Using Cuckoo Sandbox | IEEE Conference Publication | IEEE Xplore
- [3] MalwareBazaar. (n.d.). Free automated malware analysis platform. Retrieved from <https://malwarebazaar.com/>
- [4] M. Vasilescu, L. Gheorghe and N. Tapus, "Practical malware analysis based on sandboxing," 2014 RoEduNet Conference 13th Edition: Networking in Education and Research Joint Event RENAM 8th Conference, Chisinau, Moldova, 2014, pp. 1-6, doi: 10.1109/RoEduNet RENAM.2014.6955304. Practical malware analysis based on sandboxing | IEEE Conference Publication | IEEE Xplore
- [5] S. Jain, T. Choudhury, V. Kumar and P. Kumar, "Detecting Malware & Analysing Using Sandbox Evasion," 2018 International Conference on Communication, Computing and Internet of Things (IC3IoT), Chennai, India, 2018, pp. 111-116, doi: 10.1109/IC3IoT.2018.8668149. Detecting Malware & Analysing Using Sandbox Evasion | IEEE Conference Publication | IEEE Xplore
- [6] Greamo, C., & Ghosh, A. (2011). Sandboxing and virtualization: Modern tools for combating malware. *IEEE Security & Privacy*, 9(2), 79-82.
- [7] Miwa, S., Miyachi, T., Eto, M., Yoshizumi, M., & Shinoda, Y. (2007, August). Design and Implementation of an Isolated Sandbox with Mimetic Internet Used to Analyze Malwares. In *DETER*.
- [8] Tumkur, S. D., Eswarakrishnan, V., Wairagade, A., Aslam, M. A., Bilal, M., & Cheema, A. M. (2025). Optimizing Malware Detection in Virtual Cloud Environments Using Hybrid Machine Learning Approach. *Arabian Journal for Science and Engineering*, 50(23), 19463-19494.
- [9] Chauhan, A., Ailawalia, P., & Mehra, C. (2024). Advancements in Automated Malware Analysis Techniques: A Comprehensive Study. *Grenze International Journal of Engineering & Technology (GIJET)*, 10.
- [10] Shalaginov, A., Banin, S., Dehghantanha, A., & Franke, K. (2018). Machine learning aided static malware analysis: A survey and tutorial. *Cyber threat intelligence*, 7-45.
- [11] Vidarthi, D., Kumar, C. S., Rakshit, S., & Chansarkar, S. (2019). Static malware analysis to identify ransomware properties. *International Journal of Computer Science Issues (IJCSI)*, 16(3), 10-17.
- [12] Sihwail, R., Omar, K., & Ariffin, K. Z. (2018). A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis. *Int. J. Adv. Sci. Eng. Inf. Technol*, 8(4-2), 1662-1671.
- [13] 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), 1-42.
- [14] Prayudi, Y., & Riadi, I. (2015). Implementation of malware analysis using static and dynamic analysis methods. *International Journal of Computer Applications*, 117(6).
- [15] Zhang, Z., Qi, P., & Wang, W. (2020, April). Dynamic malware analysis with feature engineering and feature learning. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 34, No. 01, pp. 1210-1217).
- [16] Gandotra, E., Bansal, D., & Sofat, S. (2014). Malware analysis and classification: A survey. *Journal of Information Security*, 5(2), 56.
- [17] Oki, G. E., & Sadiq, E. W. (2025). Virtual Machines vs. Containerized Environments: A Comparative Study for Malware Analysis. *Saudi J Eng Technol*, 10(4), 206-210.
- [18] Or-Meir, O., Nissim, N., Elovici, Y., & Rokach, L. (2019). Dynamic malware analysis in the modern era—A state of the art survey. *ACM Computing Surveys (CSUR)*, 52(5), 1-48.
- [19] Liu, L., Wang, B. S., Yu, B., & Zhong, Q. X. (2017). Automatic malware classification and new malware detection using machine learning. *Frontiers of Information Technology & Electronic Engineering*, 18(9), 1336-1347