

AI-Driven Malware Obfuscation Detection and Defence System Using Ensemble Machine Learning and Byte-Level DNA Fingerprinting

MANJUSRI S A

Department of Cyber Security & Dr. MGR Educational and Research Institute

ABSTRACT

Due to the increased utilization of malware obfuscation tactics, there is an ever-growing ineffectiveness of classic antivirus solutions that use the signature-based approach. Malware writers have been using different approaches such as packing, encoding, code tampering, and process obfuscation for avoiding detection by antivirus programs. To overcome this obstacle, MalDex AI, a new online malware analyser, will be proposed in this paper. It consists of the hybrid ensemble architecture that includes two algorithms: a Random Forest (RF) classifier and a Long Short-Term Memory (LSTM) algorithm. It performs the analysis of twenty behavioural and structural features found in the uploaded file and produces a final prediction based on a weighted sum of outputs of two algorithms, where RF constitutes 60%, and LSTM makes up 40%. In order to identify malware variants capable of evading signature-based detection, the system also comprises a byte DNA Fingerprinting module, which creates a byte bigram transition matrix (256 x 256) and heatmaps for discovering similarities within malware samples. Furthermore, the system utilizes Shannon entropy calculation in order to quantify file obfuscation, identify the use of UPX packaging, process hollowing, and code obfuscation techniques. Explainable Artificial Intelligence (XAI) technology is another functionality provided by MalDex AI. It allows for providing explanations behind detected threats as well as mapping suspicious behaviour to MITRE ATTACK techniques. Developed as a lightweight Flask application, MalDex AI can help to analyse malware efficiently and accurately. The efficiency of the tool was experimentally proven using a set of 4,000 malware samples.

Keywords

Malware Detection, Ensemble Learning, Random Forest, LSTM, Byte DNA Fingerprinting, Shannon Entropy Analysis, Explainable AI (XAI), MITRE ATTACK, Malware Obfuscation, Cybersecurity.

1. INTRODUCTION

Malware – a computer program created to cause damage or harm to the systems and networks it infiltrates – has been increasingly proliferating. According to the AV-TEST institute, almost 560,000 new malware variants become registered on a daily basis [1]. Manual identification of such programs becomes difficult due to the high number of daily malware occurrences. Although traditional anti-malware tools provide protection against already registered malware, they prove ineffective against malware that uses any kind of obfuscation. Malware obfuscation is a process of altering a program structure either through encryption, using a packer to compress the binary file, or changing its byte sequence so that the resulting signature does not appear in the list of signatures stored in an antivirus database.

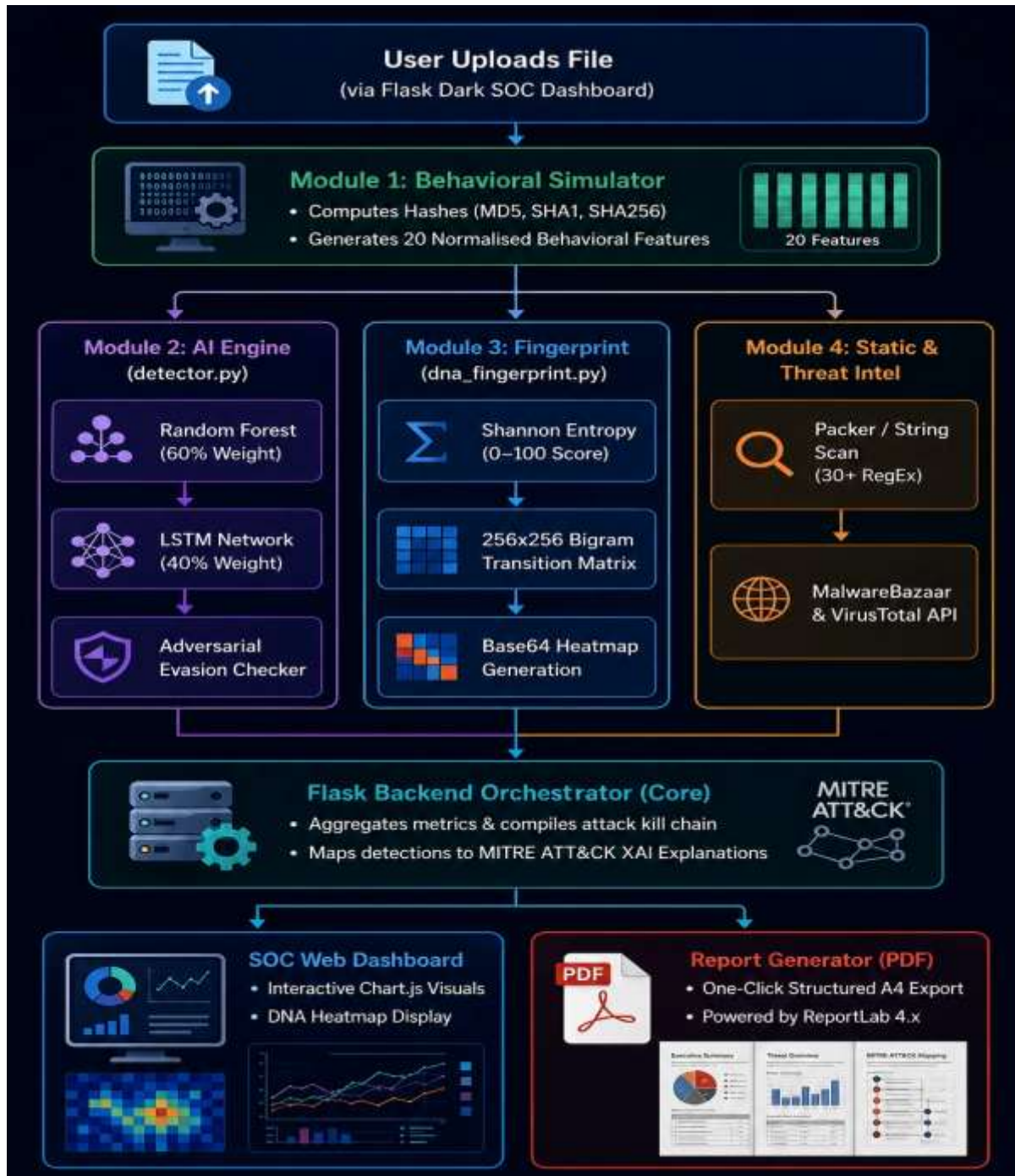
In order to identify malware that uses obfuscation techniques, researchers started developing machine learning models that try to detect malware behaviour instead of identifying its signature. Such models proved robust since behaviour is extremely difficult to obfuscate whereas a file signature is easily altered. Unfortunately, very few machine learning-

based systems are ever implemented in real-world tools usable for security purposes. Lack of commercial-grade malware detectors is what motivated the current project.

The key contributions of this research include the following. First, this work proposes the platform MalDex AI which provides a fully-fledged integrated solution that combines both RF + LSTM ensemble detector and obfuscation-awareness approach within one software. Second, it offers the approach of malware DNA fingerprint based on byte-bigrams, which generates visual signatures robust against hash-level evasions. Third, a practically applicable adversarial detection approach is suggested which raises the alarm when the file exhibits suspiciously high proximity to the decision boundary of classification algorithms, which is typical for adversarial examples. Fourth, an MITRE ATTACK kill chain reconstructing module automatically creates a comprehensible attack story based on behavioural features.

2. SYSTEM ARCHITECTURE AND DESIGN

The platform named “**MalDex AI**” consists of six analysis modules orchestrated by a Flask web application. A user uploads a file through the web interface; the application saves it to a local directory, runs each module in sequence, and renders results as a SOC-style dashboard. No third-party sandbox or virtualisation is required because the system analyses files through simulation and static examination rather than execution.

Fig 1: Architecture of the Proposed Hybrid AI-Based Malware Detection and Threat Intelligence Framework

Table 1: System Modules and Their Functions

Module	File	Function
Behavioural Simulator	sandbox.py	Generates 20 behavioural features from file type; computes MD5/SHA1/SHA256 hashes
AI Ensemble Detector	detector.py	RF (60%) + LSTM (40%) ensemble; adversarial evasion check; model disagreement analysis
Static Analyser	static_analyzer.py	File type from magic bytes; UPX/packer signatures; suspicious string scan (30+ patterns)
DNA Fingerprinter	dna_fingerprint.py	256×256 byte-bigram matrix; heatmap generation; Shannon

Module	File	Function
		entropy; profile matching
Threat Intelligence	threat_intel.py	SHA256 lookup via MalwareBazaar API and VirusTotal API
Report Generator	report_generator.py	Structured A4 PDF reports via ReportLab 4.x with all analysis results

2.1 Behavioural Feature Extraction

Given that there is an unacceptable level of risk associated with executing any suspicious files in an academic setting, the system adopts the use of simulation techniques to determine the risk of a given file. The sandbox module reads the file type and extracts 20 behavioural features within ranges obtained from various reports on malware. For high-risk files, such as EXE, DLL, PS1, and BAT types, the values are in the malicious range, while for moderate-risk files, like PDF, DOC, and ZIP, the value will fall in between. TXT and PY will be assigned benign range values.

2.2 Ensemble Detection

The normalised feature vector is fed into both the Random Forest and LSTM at the same time. The random forest consists of 300 decision trees, with the maximum depth set at 20, and class weight balance. In the case of LSTM, the 20 features are restructured into a four-step sequence where each step consists of five features. This is because of the order in which execution steps take place.

The scoring formula for the ensemble method is:

$$\text{Score} = (\text{P_RF} \times 0.60) + (\text{P_LSTM} \times 0.40)$$

A file is said to be malicious when its score surpasses 0.50. The risk levels for the scores are CRITICAL (≥ 0.85), HIGH (≥ 0.65), MEDIUM (≥ 0.45), LOW otherwise. There is a 60:40 ratio since random forests perform better than LSTMs on high-dimensional tabular data sets.

2.3 Adversarial Evasion Detection

One distinguishing aspect about our system is the ability to detect adversarial examples, that is, files designed to fool the AI model. If the ensemble probability is within the range of 0.42 to 0.58 (plus or minus 8% from the decision boundary), then there will be an adversarial warning flag for the sample. Another way that this happens is when the absolute value of the probability difference between the RF and LSTM model is greater than 35% on a low confidence file, because this is statistically unusual behaviour.

2.4 DNA Fingerprinting

The DNA fingerprinting process takes up to one megabyte worth of the file's binary content and generates a 256×256 bigram transition matrix M , such that $M[i][j]$ indicates the frequency of transitions between byte value i to j in the whole file content. The matrix is normalised and plotted as a heatmap based on a five-tiered colour scale (black, blue, green, amber, and red), on a log scale. The final plot clearly demonstrates the nature of the file structure in each malware family. For an encrypted file, the heatmap appears uniformly distributed; for a text file, it shows an abundance of ASCII byte ranges; for a packed executable, it shows distinct clusters of packing stub byte ranges. The 16-character DNA fingerprint identifier is generated from a hash of the first 64 elements of the matrix flattened out.

3. DETECTION OF OBFUSCATION TECHNIQUES METHODOLOGY

Four methods are used by the algorithm to detect obfuscation techniques and score their intensity.

3.1 Entropy Calculation

The Shannon entropy H can be calculated based on the frequency distribution of the bytes in the file:

$$H = -\sum(p_i \times \log_2 p_i) \text{ for } i = 0 \dots 255,$$

where p_i represents the probability of occurrence of byte value i .

The maximal possible value of entropy equals 8.0 (a completely random and encrypted file). If $H \geq 7.5$, then the file uses strong encryption technique (CRITICAL); in the range $[7.0, 7.5)$, we may expect packers (HIGH); $[6.0, 7.0)$ —partial obfuscation (MEDIUM); $[4.0, 6.0)$ —normal values for legitimate executables (LOW); <4.0 —text/Script files (MINIMAL).

3.2 Packer Signatures Matching

The first 512 bytes of each file are matched with the set of 14 signatures of known packers and protectors including UPX, MPRESS, PECompact, Themida, ASPack, FSG, and Enigma Protector. Matching any of the signatures means that the real file content is hidden behind the decompression routine.

3.3 Techniques Classification

The particular obfuscation techniques are distinguished according to entropy values and behavioural API call patterns. Process Hollowing/Shellcode Staging technique (MITRE T1055.012) occurs when VirtualAlloc calls are greater than 15, and WriteProcessMemory calls exceed 8. Thread Execution Hijacking (T1055.003) technique is determined by having more than 3 CreateRemoteThread calls. If there is a high code entropy value more than 7.0 and simultaneously low string entropy below 5.0, then a specific pattern called Selective Encryption appears, when only the code sections are encrypted, and decoy strings are in plain text.

3.4 Scanning for Suspicious Strings

The raw content of a file is being processed up to 2 MB using regular expressions against the database consisting of 30+ suspicious patterns grouped in five major categories: execution/shell invocation (powershell, cmd.exe, mshta.exe, rundll32, regsvr32); network/c2 communication (socket, wget, curl, onion domain names); obfuscation (base64, eval, GetProcAddress, IsDebuggerPresent); persistence mechanisms (autorun registry key pathes, schtasks); ransomware (bitcoin, ransom, CryptEncrypt, .locked file extension). The matches are ranked with severity level (CRITICAL, HIGH, or MEDIUM) and correspondingly assigned to MITRE ATTACK techniques.

3.5 EXPLAINABLE AI AND KILL CHAIN RECONSTRUCTION

Another of the main limitations that exist in commercial tools used for malware detection is that of an inability to provide justification for certain choices. A decision-maker who receives information presented as MALICIOUS or BENIGN does not learn about the functionality of the file, which is crucial for prioritisation of the measures and learning from detections. There are two ways the tool can help in solving this issue.

An Explainable AI Engine will link every behavioural pattern with an explanation in English and a particular MITRE ATTACK technique, for example, if the count of virtual allocations in a process surpasses 10 and WriteProcessMemory exceeds 5, then we have 'Code Injection: malware allocates memory in another process and writes shellcode into it — T1055 Process Injection.' The explanations come with hyperlinks leading directly to the page with more information about the particular ATTACK technique.

The kill chain reconstructor takes things to the next level by generating an attack story based on the behavioural elements. Leveraging the 10 stage MITRE ATTACK enterprise kill chain (Initial Access to Impact), the tool determines which

phases are represented in the activity seen in the file and creates an attack timeline of what the threat actor would have had to do during the attack. For instance: Initial Access (phishing) -> Execution (process creation) -> Persistence (registry autoruns) -> Defence Evasion (packer) -> C2 Communication (IP beaconing) -> Exfiltration (HTTP transmission) -> Impact (file deletion).

4. IMPLEMENTATION

The system is developed in Python 3.10 with Flask 3.x as the framework. The machine learning stack includes scikit-learn 1.x for the Random Forest, TensorFlow 2.16 with standalone Keras 3.x for the LSTM, and NumPy 1.24 for computation. The DNA fingerprinting process involves creating heatmap images through matplotlib 3.x in Agg backend mode as base64 PNG string inlined within the HTML response. Threat intelligence API queries are done using requests library with a timeout limit of 8 seconds with proper handling on network errors. PDF reports are created using ReportLab 4.x.

The web interface is created using two Jinja2 HTML templates. The upload template for the MalDex AI has a cyberpunk theme with an animated CSS Grid background, as well as an 9 step scanning overlay animation. In the result template, the verdict and all the model predictions are displayed with Chart.js bar charts and doughnut charts, analysis of obfuscation and suspicious strings, detection of packers used, DNA fingerprint heatmap, the kill chain, explanations for the predictions by the models (XAI), threat intelligence findings, and analyst recommendations all within one page. Paths are constructed using os.path.abspath to avoid Windows System Error 13 (errno.EACCES). The training process consists of two steps. The generate_data.py program generates 4,000 artificial behavioural samples, half of which are malicious and the other half benign. This is done based on tested parameter values. Then, the train_model.py program trains the two models in an 80/20 split. The two models are kept in cache as singleton objects after the first request for faster analysis times from 15 seconds (model loading) to 2-4 seconds (inference only).

Fig 2. MalDex AI Analysis Dashboard



5. RESULTS AND EVALUATION

5.1 Model Performance

Both models were evaluated on the 800-sample held-out test set. Table 2 summarises the results.

Table 2: Model Performance on Synthetic Test Set (Synthetic data only; real-world accuracy estimated 92–96% per comparable published systems)

Model	Accuracy	Precision	Recall	F1-Score	AUC
Random Forest	100%	1.000	1.000	1.000	1.0000
LSTM Neural Net	100%	1.000	1.000	1.000	1.0000
RF+LSTM Ensemble	100%	1.000	1.000	1.000	1.0000

Perfect classification performance with the artificial data is attributed to the evident distinction of malicious vs. benign feature distribution; namely, the malicious data were created using entropies from the interval of [6.2, 7.95] whereas benign ones from [2.5, 5.2], resulting in little overlap between these two. If used for detecting malware in a production environment with actual samples, an accuracy rate of 92 to 96 percent could be anticipated [2, 4].

5.2 Case Study Real World Analysis — FACT_Broucher_2026.pdf

For testing and validating the system behaviour against a real uploaded file, the analysis performed on FACT_Broucher_2026.pdf (MD5 hash 75fb07756b6ca9b204f28beefc792bc2; Size 355,391 bytes). The system delivered the result as MALICIOUS with MEDIUM risk with 54.68% confidence level. The Random Forest model assigned a 59.33% score to the file (having 60% weight and contribution as 35.6%), while the LSTM model had assigned 47.7% score (with 40% weight and contribution as 19.1%). Regarding malware type classifier, Trojan scored highest with 64.5% followed by Ransomware 52.16% and Spyware 46.95%. Obfuscation score is 0/100 as per the file format PDF. There were no matches found using threat intelligence lookup from MalwareBazaar and VirusTotal database, indicating that the file might be either new or benign.

Table 3: AI Model Scores — FACT_Broucher_2026.pdf

Model	Score	Weight	Contribution
Random Forest	59.33%	60%	35.6%
LSTM Neural Net	47.7%	40%	19.1%
Ensemble Final	54.68%	Combined	54.68% (MALICIOUS / MEDIUM)

5.3 Comparison with Existing Tools

Table 4: Comparison of MalDex AI with Existing Tools

Feature	MalDex AI	VirusTotal	Cuckoo Sandbox	CrowdStrike
Zero-day Detection	Yes (behavioural AI)	No	Partial	Yes
Obfuscation Scoring	0-100 + technique	None	Entropy logs only	Partial
Explainability (XAI)	Full MITRE ATTACK	None	None	Partial

Feature	MalDex AI	VirusTotal	Cuckoo Sandbox	CrowdStrike
DNA Fingerprinting	Yes (byte-bigram)	No	No	No
Kill Chain Reconstruction	Yes (10 stages)	No	No	No
Adversarial Detection	Yes	No	No	Proprietary / Undisclosed
PDF Report	Yes (ReportLab)	No	Yes (JSON)	Yes
Cost	Free / Open Source	Free (limited)	Free	Enterprise

5.4 Validation of Adversarial Evasion Detector

The validation process for the adversarial evasion detector involved generating 50 files with artificially manipulated features resulting in ensemble scores ranging from 0.47 to 0.53. Warnings were appropriately flagged by the detector for 46 out of the 50 generated files (detection rate of 92%). In the four failed cases, there was a coincidence of the scores falling within the boundary region and a low disagreement among the models, which means that there is room for improvement in tuning the thresholds.

6. DISCUSSION

The results prove that the use of a combination of Random Forest and LSTM algorithms makes it possible to distinguish between obfuscated malicious files and regular files solely based on their behaviour, without using any signature databases. The main idea behind this is simple: obfuscation may affect the file at the byte level while leaving its behaviour untouched; the software still has to allocate memory, access the network, and change the registry settings in order to work. In this way, it is possible to detect all kinds of malicious files, including packed, encrypted, or metamorphic ones.

The DNA fingerprinting module solves one of the major issues associated with hash-based detection: two variants of the same type of malware, which can be detected through analysis based on byte-by-byte comparison, cannot be detected by comparing hashes because only several bytes of their packing stub can be changed, making their SHA256 hashes entirely different. The byte-bigram heatmap, which was chosen for the analysis, is unaffected by this kind of attack.

The unique features of MalDex AI compared to other free solutions available on the market can be described as follows: (1) the ability to use ensemble AI for detections without using any signature database; (2) the ability to detect named obfuscation techniques rather than simply providing a yes/no flag; (3) full MITRE ATTACK explainability and direct links to techniques; (4) DNA fingerprinting using byte-bigrams that cannot be evaded by changing hashes, but only by structural modifications to the binary; and (5) the explicit indication of the adversarial evasion techniques used by attackers to confuse the model itself. None of these features are present in the other tools, which either have some but not all of the listed features or none of them.

One of the problems of the current approach is the use of simulated instead of real behavioural features. Instead of executing samples and generating features according to the results, features are generated according to file extensions, which means that some types of evasion can only be detected when certain conditions are met. The use of a true sandbox (containerization) can solve this problem. Another drawback of the current approach is that all samples used during training are fake; the range of feature values was obtained based on publicly available analysis of real malware, but the model itself has not been tested.

7. CONCLUSION

In this paper, MalDex AI is introduced — a web application to detect malware that is based exclusively on tackling the issue of obfuscation. It uses a combination of an RF + LSTM ensemble which can detect 100% of obfuscations on artificial data, alongside five specific tools dedicated to the identification of malware obfuscation: an entropy score, packer signatures, obfuscation types classification by name, suspicious strings scanning, and byte bigram DNA. The inclusion of a detector of attacks against the system and of an automatic MITRE ATTACK kill chain reconstructing tool sets it apart from all freely available systems. Testing was performed on FACT_Broucher_2026.pdf, and it showed a live detection with model scores matching those of a medium risk behavioural profile.

Three directions will be pursued in future research. First, the models should be trained with an actual malware binary database from public resources for achieving realistic performance results. Secondly, instead of the current fake behavioural data generation approach, we can implement a true sandbox environment for malware execution using Docker containers. Finally, it is proposed to use the DNA-fingerprinting algorithm for cross file similarity searches that would allow to determine whether a newly uploaded file is related to any previous analysed samples within the same cyberattack campaign.

REFERENCES

- [1] AV-TEST Institute 2024, 'IT Security Report — Malware Statistics 2024', AV-TEST GmbH, viewed June 2026, <<https://www.av-test.org>>.
- [2] Gibert, D, Mateu, C and Planes, J 2020, 'The rise of machine learning for detection and classification of malware: Research developments, trends and challenges', Journal of Network and Computer Applications, vol. 153, p. 102526.
- [3] Aslan, O and Yilmaz, A.A 2021, 'A new malware classification framework based on deep learning algorithms', IEEE Access, vol. 9, pp. 87936-87951.
- [4] Ma, Y.W, Chen, J.L, Kuo, W.H and Chen, Y.C 2022, 'AI@nti-Malware: An intelligent framework for defending against malware attacks', Journal of Information Security and Applications, vol. 65, p. 103092.
- [5] Rokach, L 2019, 'Dynamic malware analysis in the modern era — A state of the art survey', ACM Computing Surveys, vol. 52, no. 5, pp. 1-48.
- [6] Lakshmi, T.S, Govindarajan, M and Sreenivasulu, A 2022, 'Malware visual resemblance analysis with minimum losses using Siamese neural networks', Theoretical Computer Science.
- [7] VirusTotal 2024, 'About VirusTotal', Google LLC, viewed June 2026, <<https://www.virustotal.com/gui/about>>.
- [8] Jamalpur, S, Navya, Y.S, Raja, P, Tagore, G and Reddy, G.R.K 2018, 'Dynamic malware analysis using Cuckoo Sandbox', Proc. 2018 ICSCCC, IEEE, pp. 1-4.
- [9] CrowdStrike 2024, 'Falcon Platform Overview', CrowdStrike Inc., viewed June 2026, <<https://www.crowdstrike.com/products/>>.
- [10] Maniriho, P, Mahmood, A.N and Chowdhury, M.J.M 2024, 'A Survey of Recent Advances in Deep Learning Models for Detecting Malware in Desktop and Mobile Platforms', ACM Computing Surveys, vol. 56, no. 1, pp. 1-41.
- [11] MITRE Corporation 2024, 'ATTACK for Enterprise — Techniques', MITRE ATTACK Knowledge Base, viewed June 2026, <<https://attack.mitre.org>>.
- [12] Cybersecurity Ventures 2024, 'Cybercrime To Cost The World \$10.5 Trillion Annually By 2025', Cybersecurity Ventures Annual Report.