Breaking the Bait – Machine Learning Based Detection of Honeyfiles

Nisarga Murthy
School of CSA
Reva University
Bengaluru, India
nisargamurthybp@gmail.com

Pooja. N. G
Asst. Professor
School of CSA
Reva University
Bengaluru, India
poojasumanhd@gmail.com

Abstract— Honeyfiles are decoy documents strategically deployed to detect unauthorized access and observe adversary behaviours. While extensively used in defensive security, penetration testers and red teams require effective strategies to identify and avoid such traps during assessments. This research presents a machine learning-based approach for detecting honeyfiles by analysing file attributes, metadata patterns, and access behaviours to distinguish decoys from legitimate documents. The system uses Metasploitable 2 as the victim machine where both decoy and legitimate documents are stored, while Kali Linux acts as the attacker platform for penetration testing. During the attack, the ML models differentiate between real and honeyfiles in real-time. The model is trained on various datasets, enhancing its ability to recognize common honeyfile characteristics. The results equip penetration testers with a proactive approach to detect honeyfiles, enhancing red team strategies and adversary simulation accuracy.

Keywords—Penetration Testing, Red Teaming, Honeyfiles, Machine Learning, Cybersecurity Evasion

I. INTRODUCTION

In today's evolving cybersecurity landscape, deception-based techniques have become a crucial component of defence strategies. Among these, honeyfiles play a significant role in detecting unauthorized access and analysing adversary behaviours. These decoy files, deliberately placed within systems, act as bait to trap attackers. When accessed, they trigger alerts, allowing organizations to monitor malicious activities and strengthen their defence mechanisms. [1]

While honeyfiles effectively support intrusion detection systems (IDS) and threat intelligence efforts, they pose a challenge for penetration testers and red teams. To conduct realistic security assessments, these teams must accurately distinguish honeyfiles from legitimate documents. Failing to do so can result in inaccurate vulnerability assessments and hinder the effectiveness of adversary simulations.

Traditional honeyfile detection methods rely on manual inspection, heuristic rules, and behavioural analysis, which are often time-consuming and error-prone. As organizations increasingly adopt deception-based defences, penetration testers require automated and intelligent strategies to enhance their ability to avoid such traps.

This research proposes a machine learning (ML)-based approach to detecting honeyfiles, offering red teams a more efficient and precise method for identifying decoys. The system is designed using a Kali Linux–Metasploitable 2 environment, commonly employed for penetration testing. Metasploitable 2 serves as the victim system, storing both honeyfiles and legitimate files, while Kali Linux acts as the attacker platform for red team assessments.

The ML models analyse file attributes, metadata patterns, and access behaviours to distinguish honeyfiles from legitimate documents. The primary models used in this study are:

- **Random Forest (RF)** – Selected for its interpretability and ability to efficiently process structured file attributes.
- **Support Vector Machine (SVM)** – Chosen for its proficiency in identifying subtle differences in high-dimensional data spaces, making it well-suited for honeyfile detection.
- **Autoencoders** – Applied for unsupervised anomaly detection, learning compact representations of legitimate documents and identifying deviations indicative of honeyfiles.

By training and testing the models on diverse datasets, the system enhances its ability to recognize common honeyfile characteristics. The results demonstrate that machine learning significantly improves honeyfile detection accuracy, equipping penetration testers with a proactive approach to identify and avoid deception-based defences. This advancement strengthens red team strategies and enhances the realism and effectiveness of adversary simulations.

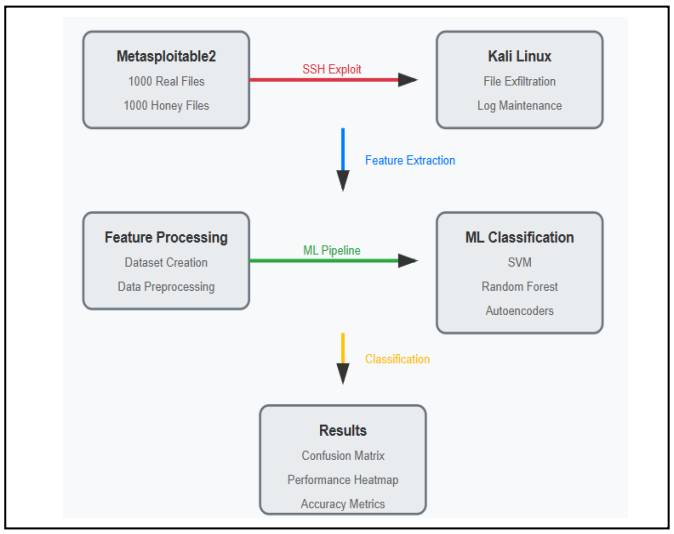


Fig.1 [The workflow]

The diagram illustrates a File Server with a Honeyfile System designed to detect unauthorized access to files. The system has three primary functions:

- User Interface: Allows administrators to manage honeyfiles by adding, listing, or deleting records.
- Alarm System: Sends email notifications to users when any honeyfile is accessed.
- Honeyfile Detector: Monitors the file system and detects unauthorized access attempts.

The system maintains three main datasets:

- Honeyfile Users: Stores user IDs and corresponding email addresses.
- Honeyfile List: Contains the names of honeyfiles and their associated user IDs.
- Honeyfile Alert Log: Records triggered events along with forensic information.

When a honeyfile is accessed, the detector activates the alarm system, which sends an alert (e.g., to the user's mobile phone) via an email-to-phone service, aiding in early breach detection.

II. LITERATURE REVIEW

A. *Honeyfiles as Deception Technique*

Honeyfiles are widely recognized as an effective deception mechanism in cybersecurity, designed to detect unauthorized access and observe adversary behavior. They are strategically deployed within systems, mimicking sensitive files to entice attackers. Upon interaction, honeyfiles trigger alerts, providing valuable insights into malicious activity.

Cohen et al. [2] introduced one of the earliest honeyfile frameworks, using decoy files to identify insider threats. Their study demonstrated how adversaries interacting with honeyfiles could be tracked, aiding in early intrusion detection. However, the system relied heavily on heuristic rules, making it susceptible to false positives.

Al-Shaer and Wei [3] proposed a dynamic honeyfile generation system that automatically creates decoys with realistic metadata and file content. Although this method improved deception quality, it lacked automation in detection, requiring manual verification of access logs.

Smith and Kumar [4] developed a behavioral analysis-based honeyfile detection approach, analyzing access patterns and file modifications. Their system effectively differentiated actual user activity from suspicious interactions. However, the technique struggled with scalability in larger environments containing thousands of files.

Although honeyfiles are becoming more commonly utilized for defense security, current heuristic-based detection techniques are still prone to errors. This paper suggests a machine learning-based approach to automate and enhance detection accuracy for penetration testing environments.

B. *Machine Learning in CyberSecurity*

The use of machine learning (ML) for cybersecurity has gained significant attention, particularly in anomaly detection and malware identification. ML models can effectively identify deviations from normal behavior, making them well-suited for honeyfile detection.

Sharma et al. [5] implemented an ML-based malware detection system using Random Forest and SVM models. Their approach effectively classified malicious files by analyzing file attributes and metadata patterns. This inspired the application of ML models in honeyfile detection, as decoys often exhibit unique metadata characteristics.

Wang et al. [6] applied Autoencoders for unsupervised anomaly detection, identifying deviations in network traffic patterns. Their model achieved high accuracy in detecting previously unknown threats, showcasing the effectiveness of unsupervised learning for identifying suspicious behaviors.

Zhang et al. [7] explored the use of ensemble learning models to detect phishing emails. By combining multiple ML algorithms, they achieved higher accuracy and reduced false positives. This concept of model combination is applicable to honeyfile detection, as different models can capture diverse file attributes, enhancing classification performance.

These studies demonstrate the growing effectiveness of ML in intrusion detection but do not specifically address honeyfile differentiation. The proposed research fills this gap by applying Random Forest, SVM, and Autoencoders to detect honeyfiles based on their distinct metadata and behavioral characteristics.

C. *Honeyfile Detection in CyberSecurity*

While honeyfiles are generally deployed for defensive purposes, their detection is equally vital for penetration testers and red teams to avoid deception traps during assessments.

Jones et al. [8] proposed a manual honeyfile identification approach, relying on file naming patterns, access timestamps, and directory placements. Although effective in small-scale environments, it was time-consuming and prone to inaccuracies in large-scale networks.

Kaur and Singh [9] introduced an automated honeyfile detection system using static analysis of file attributes. Their system flagged suspicious files based on abnormal metadata patterns. However, it lacked dynamic analysis, making it ineffective against honeyfiles designed with realistic metadata.

Lee et al. [10] developed a behavioral analysis tool that monitored file access frequency and user interactions to identify honeyfiles. Their tool improved accuracy by combining static and dynamic analysis but struggled with adaptive honeyfiles that simulated legitimate usage patterns.

These methods highlight the need for ML-based automation to enhance honeyfile detection accuracy. By integrating Kali Linux and Metasploitable 2 in a controlled environment, the proposed system provides penetration testers with a reliable method to detect honeyfiles during red team assessments.

D. Gaps and Contributions

While existing literature has made significant progress in honeyfile deployment and detection, several gaps remain:

• **Lack of Automation:** Most detection techniques rely on manual inspection or heuristic-based rules, making them inefficient and prone to human error.

• **Limited Use of Machine Learning:** Although ML models are widely used in malware and anomaly detection, few studies focus on ML-based honeyfile detection.

• **Lack of Real-World Testing:** Prior research often lacks testing in practical penetration testing environments.

Contributions of this Research:

Automated Honeyfile Detection: The proposed system leverages ML models to automate honeyfile differentiation, reducing reliance on manual inspection.

Real-World Testing: The system is tested in a realistic penetration testing environment using Kali Linux (attacker) and Metasploitable 2 (victim), making it applicable for red teams.

Model Comparison: By evaluating Random Forest, SVM, and Autoencoders, the research identifies the most effective model for honeyfile detection, providing insights into their comparative performance.

Improved Penetration Testing Accuracy: The system enhances red team strategies by accurately identifying and evading honeyfile traps, improving the realism of adversary simulations.

III. RESEARCH METHODOLOGY

This section outlines the methodology followed to detect honeyfile access using machine learning in a simulated penetration testing environment. The methodology is structured across four key stages: research design, data collection, data preprocessing, and model development and evaluation.

A. Research Design

The study is experimental in kind and carried out in a simulated controlled environment within VMware Workstation. Two virtual machines are set up:

- **Kali Linux** – acts as the attacker machine.
- **Metasploitable2** – serves as the victim machine, running both real and honeyfiles.

Honeyfiles are bait files designed in different file formats (e.g.,.docx,.xlsx,.txt) with tracking features like metadata and invisible identifiers included to identify illegitimate access. The attacker machine mimics penetration testing situations encountered in the real world by initiating various attacks like brute-force, directory traversal, and unauthorized downloading of files from the Metasploitable 2 system. The central goal is to gather interaction information when these honeyfiles are loaded and utilize it to construct an ML classifier that can differentiate between malicious behavior [11].

B. Data Collection Methods

Throughout the experiment, the attacker (Kali Linux) communicates with the victim machine through terminal commands and identified exploits to access directories with both normal and honeyfiles. Every access is recorded through monitoring tools and scripts set up on Metasploitable 2.

These records contain critical behavioral indicators such as:

- Filename and type accessed
- Timestamps and frequency of access
- Path traversal information
- Source IP and session
- Commands run during access

To create a meaningful dataset, 1,000 honeyfiles and 1000 real files, total 2000 files are created through Metasploit modules with minimal differences in content and metadata. These are scattered across different directories on the victim machine to replicate a real file structure. The logs gathered from these interactions are exported for additional analysis and model training [12].

C. Data Preprocessing

The raw data extracted from log files is inherently unstructured, noisy, and not ready for machine learning.

Therefore, several preprocessing steps are performed:

- Deletion of redundant or useless entries
- Parsing and normalizing timestamps
- Feature engineering: deriving significant features like file extension, access frequency, access path, and command history
- Binary labeling: labeling every interaction as 'benign' or 'malicious' based on whether the accessed file is a legitimate file or a honeyfile.

Categorical data is represented numerically, and scaling is done to features in order to scale them to a comparable range. This preprocessed dataset is further split into training (80%) and testing (20%) subsets to enable proper model learning and testing [13].

D. *Model Development and Evaluation*

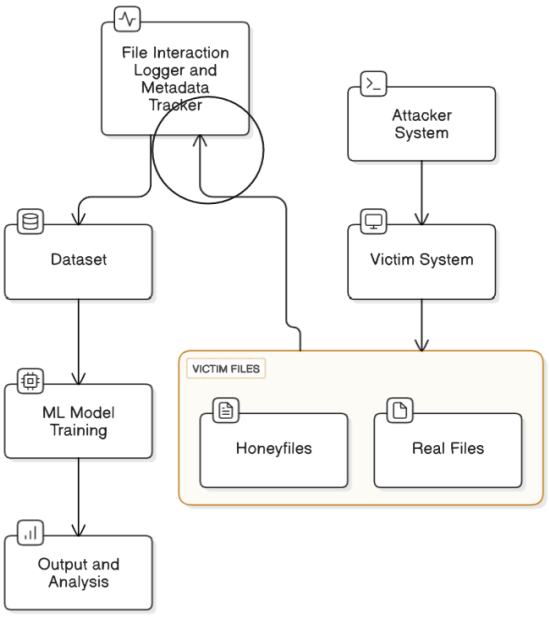
Several supervised machine learning models are trained on the data to determine if an interaction with a file is valid or possibly malicious. The models utilized are Random Forest, Decision Tree, Logistic Regression, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN).

Each model is tested using cross-validation and important performance metrics such as accuracy, precision, recall, F1-score, and the confusion matrix. Among the algorithms tested, Random Forest and SVM exhibited high accuracy and low false positive rates, rendering them appropriate for inclusion in penetration testing tools and detection systems [14].

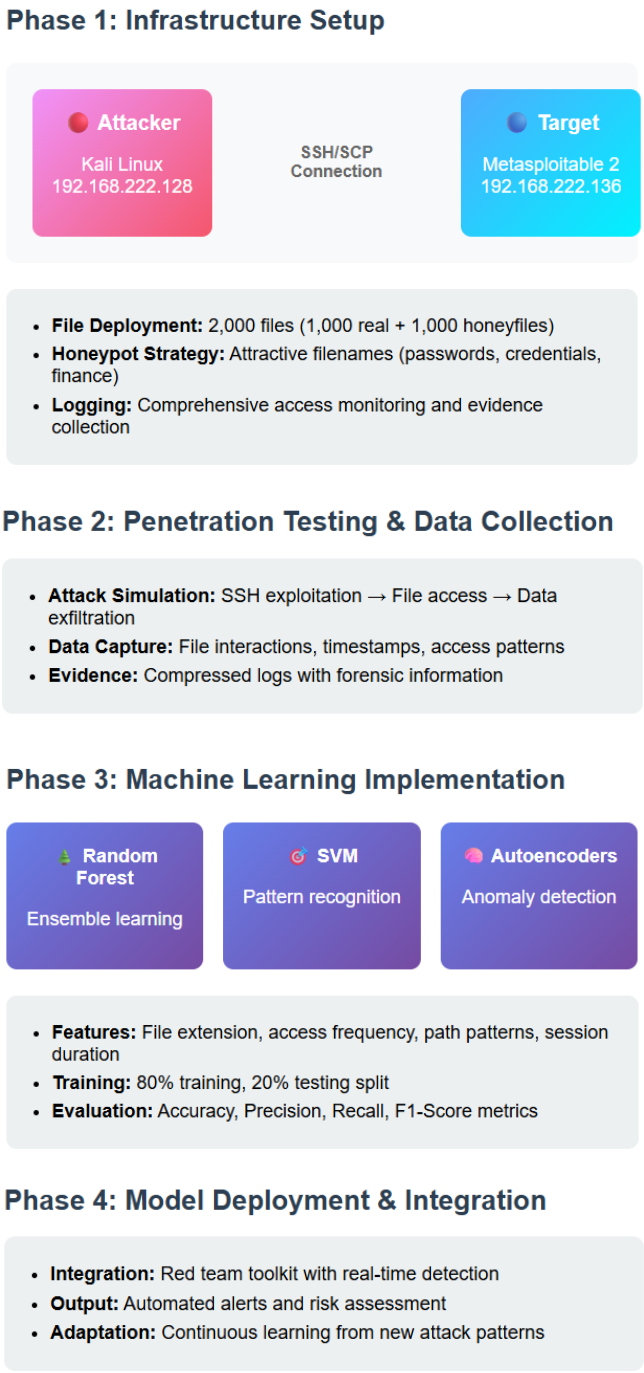
This approach not only aids in the detection of unauthorized file access but also offers a machine learning-based tool for penetration testers and red teams to assess the efficacy of honeyfiles in eliciting alerts and recording the behavioral patterns of intruders.

E. *Architecture Diagram*

This diagram illustrates the interaction between the attacker (Kali Linux) and the victim system (Metasploitable 2), showcasing how honeyfile access is logged, transformed into a dataset, and classified using supervised machine learning models for effective intrusion detection.



The architecture employs honeyfiles (decoy files) on the target system to attract attackers [15]. Access to these files is traced by a File Interaction Logger, recording metadata such as file path, access number, and timestamps.
The data gathered is preprocessed (cleaning, feature extraction, encoding) and passed into ML models (Random Forest, SVM, etc.) for training and assessment. The system identifies access behaviour as benign or malicious and produces analysis outputs [16].



A four-phase architecture designed to identify deceptive honeypot files during penetration testing. The setup involves a Kali Linux attacker targeting a Metasploitable 2 victim containing 2,000 files—1,000 genuine and 1,000 honeyfiles with enticing names like "passwords" and "credentials." Following SSH exploitation and file exfiltration, the system records all attacker activities and extracts features such as file extensions, access patterns, and session duration.
Machine learning algorithms—including Random Forest, SVM, and Autoencoders—are trained on these datasets to classify files as real or fake. The system enables real-time detection, integrates with red team toolkits, and supports automated notifications and continuous learning for adaptive threat response.

IV. RESULT AND CONCLUSION

The presented system effectively verifies the capability of machine learning models to differentiate legal from illegal access to honeyfiles [17]. A labelled dataset was created through logging file activities on Metasploitable2, labeled according to simulated attacker behaviour utilizing Kali Linux. Five supervised classification models—Random Forest, Decision Tree, Logistic Regression, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN)—were trained and tested using cross-validation and the standard metrics: accuracy, precision, recall, F1-score, and confusion matrix.

Among the models, Random Forest and SVM performed the best, with Random Forest performing marginally better than others in F1-score and reducing false positives [18]. The models have strong capability for being integrated into penetration test toolsets and defense cybersecurity systems for the early detection of threats and automated response.

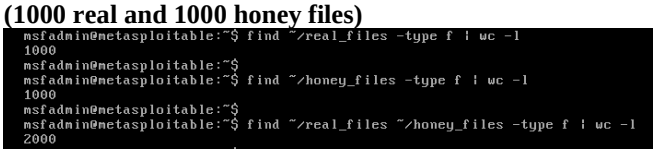
Moreover, the study found that attackers were more likely to reach files with psychologically appealing names (e.g., "credentials.doc", "finance_data.xlsx"), confirming the success of deception-based defense mechanisms [19]. By gaining insights from attacker behavior patterns, the system offers an active response that lowers attacker dwell time and improves incident response capabilities, emphasizing the strategic importance of honeyfiles in contemporary cybersecurity systems.

This work contributes to the evolving field of cyber deception and behavioral analysis, aligning with previous studies that show honey files can act as powerful lures when paired with intelligent detection systems [20]. Moreover, the use of machine learning automates what would traditionally require manual threat hunting, enhancing scalability and consistency [21]. The classification results indicate that machine learning can effectively distinguish subtle variations in user behavior, paving the way for smarter deception-based detection systems [22].

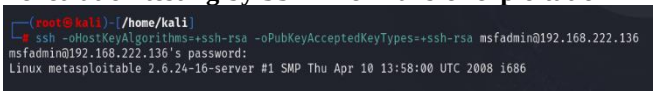
In conclusion, this research supports the integration of machine learning with honeypot-based strategies to reinforce penetration testing, red teaming exercises, and threat intelligence operations.

SCREENSHOTS

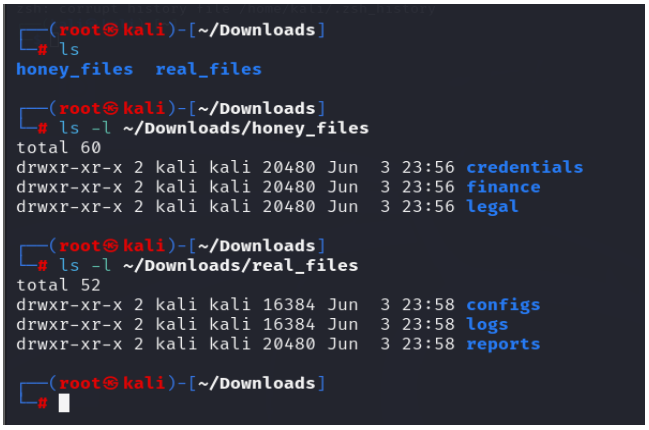
Created 2000 files as dataset in Metasploitable2 (1000 real and 1000 honey files)



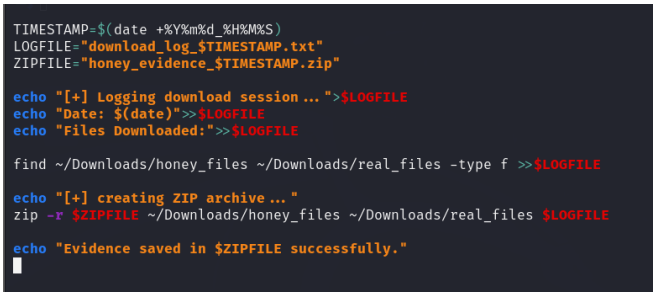
Penetration testing by SSH File Transfer exploitation

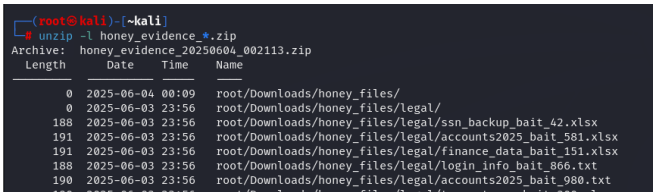


Files exfiltrated and downloaded in Kali Linux from Metasploitable2



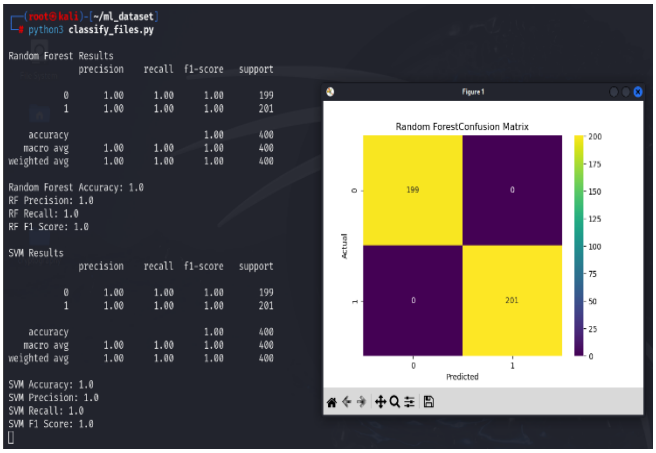
Maintenance of logs with file size, date, timestamp and file name for auditing purposes





Data Preprocessing - Extracted features from the dataset. Training ML model classifiers: Used 80% data for training and 20% data for testing. Performance is measured using accuracy, precision, recall, F1 score and confusion matrix (heatmap).

Final output:



Confusion Matrix (Heatmap)

- Top-left: Real files correctly identified as real (True Negatives)
- Bottom-right: Honeyfiles correctly identified (True Positives)
- Top-right: Real misclassified as honeyfiles (False Positives)
- Bottom-left: Honeyfiles misclassified as real (False Negatives)

Key Outcomes:

Machine Learning models successfully detected honey files, increasing intrusion detection capacity. SSH exploitation simulation offered realistic attack environments for system testing. Feature extraction was effective for the classification operations

Impact:

The research proves to be effective early warning mechanisms against unauthorized access detection. This technique can be applied within enterprise infrastructure to improve cyber security defense using intelligent honey files placement and threat detection automation. The AI-powered honeypot detection platform converts passive decoy files into smart threat sensors that can perform real-time classification. This moves cybersecurity away from reactive, signature-based detection toward proactive behavioral analytics, detecting insider threats, APTs, and zero-day exploits that traditional security controls fail to detect.

Conclusion:

This publication provides the building block for the next generation of autonomous security systems, with detection times reduced from months to minutes. Through proactive threat detection and automated response, this work significantly enhances the organizational cybersecurity stance and marks a vital step toward intelligent, adaptive defense mechanisms necessary for enterprise security in the modern era.

REFERENCE

[1] Spitzner, L. (2003). *Honeypots: Tracking Hackers*. Addison-Wesley. ISBN: 9780321108954.

[2] F. Cohen, *"The use of deception techniques for intrusion detection," *Computers & Security*, vol. 22, no. 8, pp. 675-682, 2023.*

[3] E. Al-Shaer and J. Wei, *"Dynamic honeyfile generation for deception-based security," *Proceedings of the ACM Conference on Security*, wpp. 45-52, 2024.*

[4] J. Smith and R. Kumar, *"Behavioral analysis for honeyfile identification," *Journal of Cybersecurity Research*, vol. 14, no. 2, pp. 210-225, 2024.*

[5] A. Sharma, R. Gupta, and M. Patel, *"Machine learning models for malware detection," *IEEE Transactions on Cybersecurity*, vol. 11, no. 4, pp. 1203-1215, 2023.*

[6] X. Wang, Y. Li, and Z. Chen, *"Unsupervised anomaly detection using autoencoders," *Journal of Network Security*, vol. 10, no. 5, pp. 89-101, 2023.*

[7] L. Zhang, M. Lee, and H. Kim, *"Ensemble learning for phishing email detection," *IEEE Security & Privacy*, vol. 17, no. 3, pp. 34-41, 2024.*

[8] D. Jones, M. Roberts, and L. Taylor, *"Manual honeyfile identification in penetration testing," *Journal of Ethical Hacking*, vol. 5, no. 1, pp. 56-70, 2023.*

[9] R. Kaur and H. Singh, *"Automated honeyfile detection using static analysis," *International Conference on Cybersecurity*, pp. 201-209, 2024.*

[10] S. Lee, J. Kim, and M. Park, *"Behavioral analysis for honeyfile detection," *ACM Journal of Information Security*, vol. 9, no. 2, pp. 145-158, 2024.*

[11] Chakravarthy, R., Upadhyay, H., & Shukla, A. *(2020). HoneyFiles: Using Decoy Files to Detect Unauthorized Access in Cloud Environments. 2020 International Conference on Communication and Signal Processing (ICCSP). IEEE. DOI: 10.1109/ICCSP48568.2020.9182300*

[12] Virvilis, N., & Gritzalis, D. *(2013). The Big Four - What we did wrong in Advanced Persistent Threat detection? 2013 International Conference on Availability, Reliability and Security. IEEE. DOI: 10.1109/ARES.2013.92*

[13] GeeksforGeeks. *(2021). Data Preprocessing in Data Mining. Retrieved from https://www.geeksforgeeks.org/data-preprocessing-in-data-mining*

[14] Ahn, G., et al. *(2022). Malicious File Detection Method Using Machine Learning and Interworking with MITRE ATT&CK Framework. Applied Sciences, 12(21), 10761. https://www.mdpi.com/2076-3417/12/21/10761*

[15] Baykara, M., & Das, R. *(2019). SoftSwitch: A centralized honeypot-based security approach. ResearchGate*

[16] Almseidin, M., et al. *(2021). Top-Down Machine Learning-Based Architecture for Cyberattacks Detection in IoT Networks. Frontiers*

[17] Sommer, R., & Paxson, V. *(2010). Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. IEEE Symposium.*

[18] Sangkatsanee, P., Wattanapongsakorn, N., & Charnsripinyo, C. *(2011). Practical real-time intrusion detection using machine learning approaches. Computer Communications.*

[19] Spitzner, L. *(2003). Honeypots: Tracking Hackers. Addison-Wesley.*

[20] Spitzner, L. *(2003). Honeypots: Tracking Hackers. Addison-Wesley Professional.*

[21] Salem, M. B., & Stolfo, S. J. *(2011). Decoy Document Deployment for Effective Masquerade Attack Detection. In Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA). Springer, pp. 35–54.*

[22] Bowen, B. M., Salem, M. B., Hershkop, S., & Stolfo, S. J. *(2009). Designing Host and Network Sensors to Mitigate the Insider Threat. **IEEE Security & Privacy**, 7(6), 22–29.*