# Discovering SSH Attack Patterns using Cowrie Honeypot and K-Means Clustering

Samadram Govind Singh
Central Agricultural University Imphal
Lamphelpat Imphal
Near ICAR Complex

## ABSTRACT
This paper focuses on interaction of Honeypots with Machine Learning for threat detection by finding out the patterns, anomalies, and learn from them. In this particular study, Cowrie Honeypot has been deployed on an Ubuntu Server, and its own environment is set up using python. The environment is totally isolated from the original actual server environment, and cowrie mimics the original environment, thereby luring the Hackers/Attackers to fall into the trap. Cowrie generally interacts with the SSH environment, and all the commands, IP addresses, and timestamps are captured in the log file, which is saved in the path defined by the Administrator.

Further, the log file is converted to csv file for feeding the collected data to Altair RapidMiner for its Clustering Algorithm. In RapidMiner, the csv file is retrieved, fed to Select Attribute so that the desired attributes are selected and filtered. Cowrie log generally contains a handful of noise, so normalization is needed. However, since normalization is done using z-transformation, it accepts only numerical values. This nominal-to-numerical converter is added in the process for further feeding to the Normalize operator. The normalized data is then fed to the Clustering operator, where the K-Means Clustering Algorithm is deployed in this research. In this study, 3 Clusters are studied. Using clustering analysis revealed distinct patterns in SSH honeypot attack behavior, effectively transforming unprocessed log data into actionable intelligence for strengthening proactive security responses.

In summary, integrating honeypot deception strategies with machine learning represents a significant advancement in the field of cybersecurity. This combined approach enhances threat detection and analysis while paving the way for robust, adaptive, and self-evolving security systems capable of countering ever-changing cyber threats.

## General Terms
Cybersecurity, Network Security, Machine Learning, Pattern Recognition, Data Mining.

## Keywords
Honeypots, Cowrie, Ubuntu, Machine Learning, SSH.

## 1. INTRODUCTION
In an era of rapid digital transformation and networked systems, cybersecurity concerns have grown more complex. Security experts are being compelled to continuously create new defenses due to the increasing frequency and sophistication of cyberattacks. Shyamalendu Paul et al. [1] studied the development of an AI-based Honeypot system. One promising tactic is to combine machine learning (ML) techniques with honeypot systems, which are designed to attract and deceive malicious actors.

Honeypots are decoy systems that mimic vulnerable network targets. By luring attackers into a controlled environment, they enable the collection of invaluable information about adversarial behaviors, tactics, and emerging threat trends. Iyer et al. [2] show that analysis of these interactions used to require a great deal of manual labor and experience. However, the advancement of machine learning has opened up new avenues for enhancing honeypot functionality. It is possible to analyze vast amounts of interaction data.

### 1.1 Honeypots
A honeypot is a cybersecurity technique intended to serve as a network or decoy system to entice, identify, block, or investigate attempts at unauthorized use of information systems. A honeypot is a decoy system or application intended to lure and examine malicious activities, functioning as a trap for attackers while providing valuable intelligence on emerging threats. Dakic et al. [3] describe the real-world interaction of Honeypots, analyze attack behaviour, cyber intrusion, and phishing. In order to draw attackers and give defenders a controlled environment in which to watch their behavior, honeypots are purposefully set up to look vulnerable, mimicking actual targets.

Honeypots are useful for identifying intrusions and gathering information about attack methods, resources, and goals; they don't offer real services to authorized users. After the data is sent for analysis, the pattern is created with the appropriate defenses to prevent attacks in the real world. They fall into the following categories:

**Table 1. Types of Honeypots**

| Category | Type | Description |
|---|---|---|
| Based on Interaction | Low-Interaction | Simulates limited services; easy to deploy; low risk. |
| | Medium-Interaction | Emulates more complex services; balances risk and visibility. |
| | High-Interaction | Full OS or service deployment; collects detailed attacker behavior. |
| Based on Purpose | Production Honeypot | Deployed within enterprise networks to detect and deflect attacks. |
| | Research Honeypot | Used for studying attacker methods and gathering threat intelligence. |
| Based on Role | Server Honeypot | Imitates vulnerable server services like HTTP, SSH, FTP, etc. |
| | Client Honeypot | Simulates clients to interact with malicious servers and detect threats. |
| Based on | Virtual | Deployed as virtual |

| | | |
|---|---|---|
| Architecture | Honeypot | machines for flexibility and isolation. |
| | Physical Honeypot | Runs on dedicated hardware; used for high-interaction setups. |

## 1.2 Machine Learning in Cyber Security:

Machine Learning (ML) is a subfield of artificial intelligence that enables systems to learn patterns from data and make predictions or decisions without being explicitly programmed for specific tasks.

In cybersecurity, ML is primarily used for:

- Intrusion Detection Systems (IDS)
- Anomaly Detection
- Malware Classification
- Phishing Detection
- User Behavior Analytics

ML models used in cybersecurity include:

- Supervised Learning (e.g., Random Forest, SVM, Logistic Regression): Trained on labeled data to classify threats.
- Unsupervised Learning (e.g., K-means, DBSCAN): Used to identify anomalies in unlabeled data.
- Deep Learning (e.g., CNN, RNN, LSTM): Useful for detecting complex patterns and sequences in large datasets.

**Table 2. Classification of Machine Learning**

| Category | Type | Common Algorithms | Use Cases |
|---|---|---|---|
| Supervised Learning | Classification | SVM, Decision Trees, Random Forest, Naive Bayes | Email filtering, fraud detection, disease diagnosis |
| | Regression | Linear Regression, Ridge, Lasso, SVR | Stock prediction, price forecasting, risk assessment |
| Unsupervised Learning | Clustering | K-means, DBSCAN, Hierarchical Clustering | Customer segmentation, anomaly detection |
| | Association | Apriori, Eclat, FP-Growth | Market basket analysis, recommendation systems |
| Semi-Supervised Learning | – | Variants of supervised/ unsupervised algorithms | Web content classification, medical diagnosis |
| Reinforcement Learning | – | Q-Learning, Deep Q-Network (DQN), SARSA | Robotics, game playing, autonomous vehicles |
| Self-Supervised Learning | – | Contrastive Learning, Autoencoders | Natural language processing, image recognition |

This paper explores the intersection of machine learning and honeypots to demonstrate how combining these two domains can lead to more intelligent and dynamic defenses. A study by Narayana Gaddam et al [4] shows a great development of attack activities and performance enhancement up to a visual difference after the introduction of AI-enhanced Honeypots and Machine Learning. After providing a summary of the basic concepts behind both machine learning and honeypots, the current cybersecurity problems where traditional honeypot methods fall short are examined. The discussion then turns to a number of machine learning models, such as supervised, unsupervised, and reinforcement learning models, that can be used to detect, classify, and predict malicious activity. Using a variety of case studies and experimental analyses, this work evaluates how well these models perform in honeypots, providing deeper behavioral insights and proactive threat mitigation strategies.

The goal of this research is to provide important insights into the creation of resilient, adaptive cybersecurity frameworks by combining state-of-the-art machine learning techniques with well-established decoy technologies. In the end, combining machine learning with honeypots not only improves cyber threat detection and analysis but also opens the door for more responsive and predictive security solutions, laying the foundation for the next wave of cyber defense. Narayana Gaddam et al. [4] have an in-depth report on the combination of ML with Honeypot for high-level threat detection with scalable and lightweight features.

# 2. REVIEW OF LITERATURE

Honeypots and honeynets are essential components in the field of cybersecurity, offering innovative methods for detecting, analyzing, and mitigating cyber threats. A honeypot is a purposefully weak system or network resource that is intended to look like a real target to hackers. It acts as a trap to draw in malevolent actors so that their actions can be observed without endangering actual systems. J. Franco et al [5] set up the Honeynet architecture in three Generation Phases. In contrast, a honeynet is a system of linked honeypots that replicates a full and authentic network environment.

Honeypots and honeynets contribute to the development of threat intelligence. J. Franco et al. [5] demonstrate with a survey on the role of Honeypots with Honeynet architecture. By analyzing the behavior of attackers in a controlled environment, security professionals can gather valuable data on malware variants, command-and-control (C2) infrastructures, and attacker tactics, techniques, and procedures (TTPs). Martínez S. et al. [6] demonstrate with the use of High Interaction Honeypots like Honeyd, Dioneda, and Capture-HPC. This intelligence can be used to enhance the performance of intrusion detection and prevention systems (IDS/IPS), improve patch management, and inform security policies.

However, the deployment of honeypots and honeynets is not without challenges. Sokol et al. [7] discuss the EU Laws for its deployment Privacy Policy. High-interaction systems, if not properly isolated, can be exploited as a platform to launch attacks on other networks. Additionally, Mokube et al. [8] discuss that sophisticated attackers may detect the presence of honeypots and avoid or manipulate them, reducing their effectiveness. Ethical and legal considerations also arise,

particularly when dealing with real-world malware and threat actors, as mentioned by Sokol et al. [7] where IP addresses are personal and exploiting it for personal use and analysis may lead to various Privacy Policy laws and regulations.

## 2.1 Machine Learning in Cyber Security:

Parallel to the development of honeypots is the growing body of work on machine learning (ML) applications in cybersecurity. ML's ability to process large-scale data. A study by Bharadia et al. [9] demonstrates an approach that begins with the gathering of a comprehensive dataset containing both phishing and genuine samples. Key features such as email headers, message text, and embedded URLs are then extracted, followed by training a supervised machine learning model to distinguish between them. Identifying subtle patterns and adapting to changing conditions has made it invaluable. Research has explored:

Supervised Learning: Algorithms like decision trees, support vector machines (SVM), and neural networks have been utilized for classifying known threats and detecting malware.

Unsupervised Learning: Methods such as clustering (e.g., k-means, DBSCAN) have proven effective for anomaly detection, particularly in scenarios where labeled data is scarce.

## 2.2 Convergence: Machine Learning Integrated with Honeypots:

Recent literature focuses on the symbiotic integration of ML models within honeypot environments, aiming to enhance the system's responsiveness and accuracy. This integrated approach has led to several innovative applications:

Real-Time Threat Analysis: Embedding ML models within honeypots enables real-time processing of attack data, allowing for immediate classification and response. Studies have documented systems that flag unusual patterns in real-time, significantly reducing the window of exposure.

Behavioral Fingerprinting of Attackers: Supervised learning techniques have been used to match current attack patterns with historical data, providing insight into attacker profiles and tactics. This approach aids in predictive modeling, enabling anticipatory defense measures.

# 3. METHODOLOGY

## 3.1 Installation of Cowrie

With the Ubuntu Server Environment, to install cowrie, the following command is used in terminal.

1. sudo apt update && sudo apt upgrade -y
2. sudo apt install -y git python3 python3-venv python3-pip libssl-dev libffi-dev build-essential libpython3-dev authbind

These commands will install and update Python 3 along with all necessary dependencies. The `authbind` package is included to allow non-root users to bind to low-numbered ports, such as 22 or 23, simulating a real-world environment. However, using `authbind` is optional; by default, Cowrie will be accessible through port 2222. The command "ufw allow 2222/tcp" is then added for Ubuntu firewall opening.

Following are the steps for Cowrie Configuration:

**Step – I: Create a Cowrie User:**

sudo adduser --disabled-password cowrie

A dedicated cowrie user is created, and Cowrie is a honeypot; it pretends to be a vulnerable system. The goal is to ensure that the cowrie user cannot access SSH, does not have a password, and is unable to log in directly. The --disabled-password flag does not completely lock the account; it merely prevents login using a password. However, it's still possible to use "su – cowrie" or "sudo -u cowrie" from a privileged user.

Next step is to switch to cowrie user

Switched to Cowrie: su – cowrie

**Step – II: Clone Cowrie Repository:**

git clone https://github.com/cowrie/cowrie.git

cd cowrie

**Step - III: Setup Python Virtual Environment**

1. python3 -m venv cowrie-env
2. source cowrie-env/bin/activate
3. pip install --upgrade pip
4. pip install -r requirements.txt

Python virtual environment is set up using the above command and then activated.

**Step – IV: Copy default configuration files**

1. cp etc/cowrie.cfg.dist etc/cowrie.cfg
2. cp etc/userdb.txt.dist etc/userdb.txt

cowrie.cfg is the configuration file, and the userdb.txt is the file that contains the fake user details along with the corresponding passwords. Easily predictable user and password combinations are stored as shown below:

username:password:userid
root:123456:1000
admin:admin:1001
test:test:1002

**Step – V: Run Cowrie**

Finally, after setting up the cowrie, the honeypot cowrie is run using the command "bin/cowrie start", and to view the logs, the "tail -f log/cowrie.log" command can be used. Further, the log file is saved in the cowrie/log folder and saved with extensions .log as well as .json. This .json file can be converted to csv file for further study for Machine Learning.

By default, cowrie runs on port number 2222, and the same is made open in the firewall. It can be opened by using the command "sudo ufw allow 2222/tcp" and "sudo ufw reload". Further, if there is a requirement for a change of port number, authbind can also be used.

**Step – VI: Cowrie Log**

Now that the cowrie is installed and set up successfully, the cowrie status can be checked by using the command "bin/cowrie status" from the virtual environment. Vlad-Iulius Năstase et al. [10] talk about the exploitation of cowrie ssh for better yield in threat protection. To see the real-time log "tail -f log/cowrie.log" command can be used from the virtual environment. Cowrie log is saved in .log and .json, both of which can be downloaded using the scp command from a remote location and converted back to csv file for training the data.

## 3.2 System Architecture

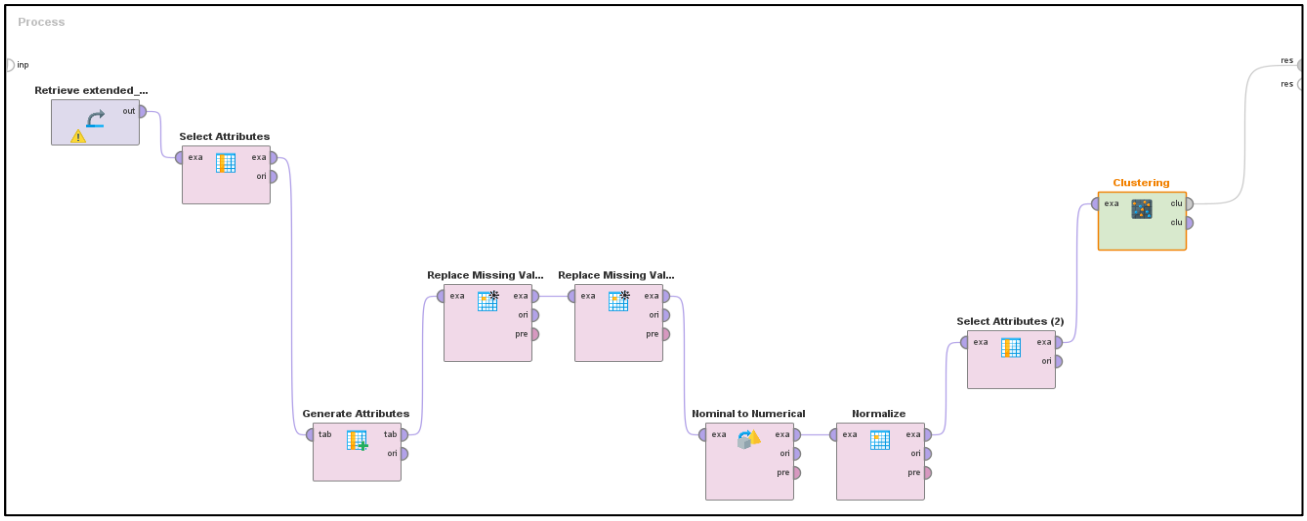


**Fig 1: Processes in K-Means Clustering**

The implementation consists of an integrated system combining high-interaction honeypots and machine learning models. The honeypot environment is deployed using Cowrie, which simulates an SSH service to attract attackers. All interaction data - including login attempts, command execution, and file transfers - is logged and transferred to a centralized server for analysis in the form of a CSV file. Patrik Krajčík et al [11] describe an overview of Cowrie's operation modes with emphasis on its simulation mode system architecture, the types of data it captures, and the key limitations identified during its deployment.

The raw data gathered from Cowrie logs underwent a thorough data preprocessing stage. This involved cleaning the data to remove inconsistencies, null entries, and duplicate records. Noise filtering was applied to eliminate irrelevant data points that could interfere with model training. A detailed study on Accuracy, Sensitivity, Precision, and False Positive Rate is well calculated in [13] Chaoyu Zhang et al. The remaining data was then structured into meaningful features such as source IP addresses, number of failed login attempts, session durations, command patterns, and request frequencies. These features were normalized to ensure consistent scaling across all input variables. The CSV file is then fed to RapidMiner for data cleaning. This process helped in identifying the most significant attributes for training effective machine learning models.

## 3.3 Data Collection and Preprocessing

The Cowrie logs are parsed to extract key features such as:

- Source IP address
- Timestamp of connection
- Command patterns
- Session duration
- Failed login attempts

timestamp, eventid, src_ip, username, password, input, url, shasum, session – these are the information extracted and fed to RapidMiner for Clustering.

In Altair RapidMiner Studio, the following steps are executed:

1. Import the Dataset
2. Repository > Right-click > Import Data.
3. Add Read CSV
4. Add Select Attributes to choose only relevant fields (e.g., eventid, src_ip, username, command, etc.).
5. Preprocess Data
   Convert categorical to numerical using:
   Nominal to Numerical: For eventid, username, input, etc.
   Assign correct types:
   timestamp: Date
   src_ip, username, password, eventid, session, url, shasum: Polynominal
   input: Polynominal
6. Select Attributes operator.
   Connect your dataset to it.
   In parameters, select only:
   eventid, src_ip, username, password, input, url, timestamp.
7. Normalize numerical data using:
   Normalize operator: (especially important for distance-based clustering like K-Means).
8. Nominal to Numerical for:
   eventid, username, password, src_ip, url.
9. Normalize (for K-Means only):
   Drag Normalize operator.
   Use default settings (Z-Transformation or Min-Max)
10. Add K-Means Operator
    Drag K-Means onto the canvas.
    Set k = 3 (you can tune this later).
    Connect the Normalize output to K-Means.

## 3.4 Model Selection and Training

**K-Means (Unsupervised):** For anomaly detection based on command pattern deviations.

The K-Means clustering Model was used for training purposes. Youguo Li et al [12] talk about traditional and improved K-Means Clustering Algorithm and their effectiveness in unsupervised learning.

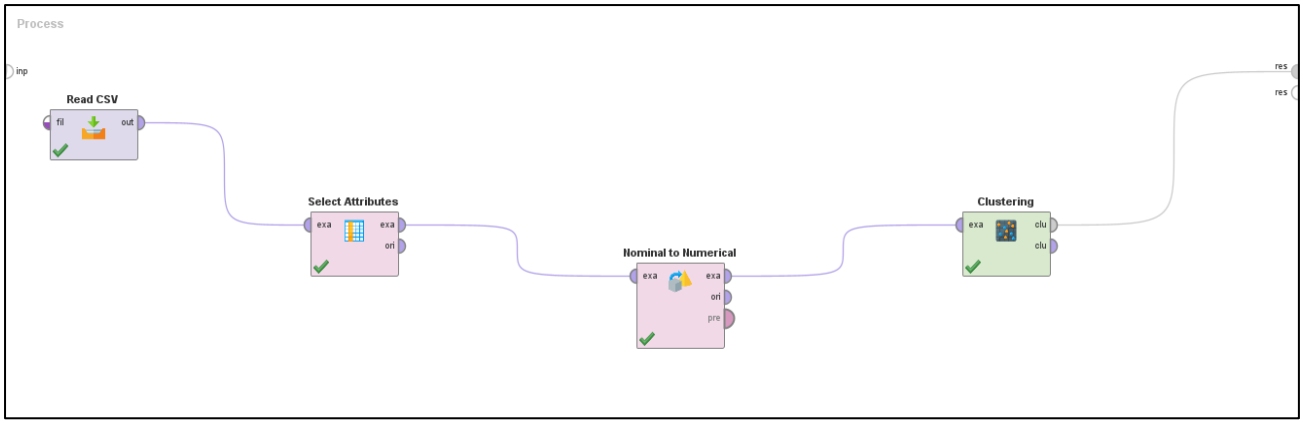


**Fig 2 Actual K-Means Clustering Process without Noise**

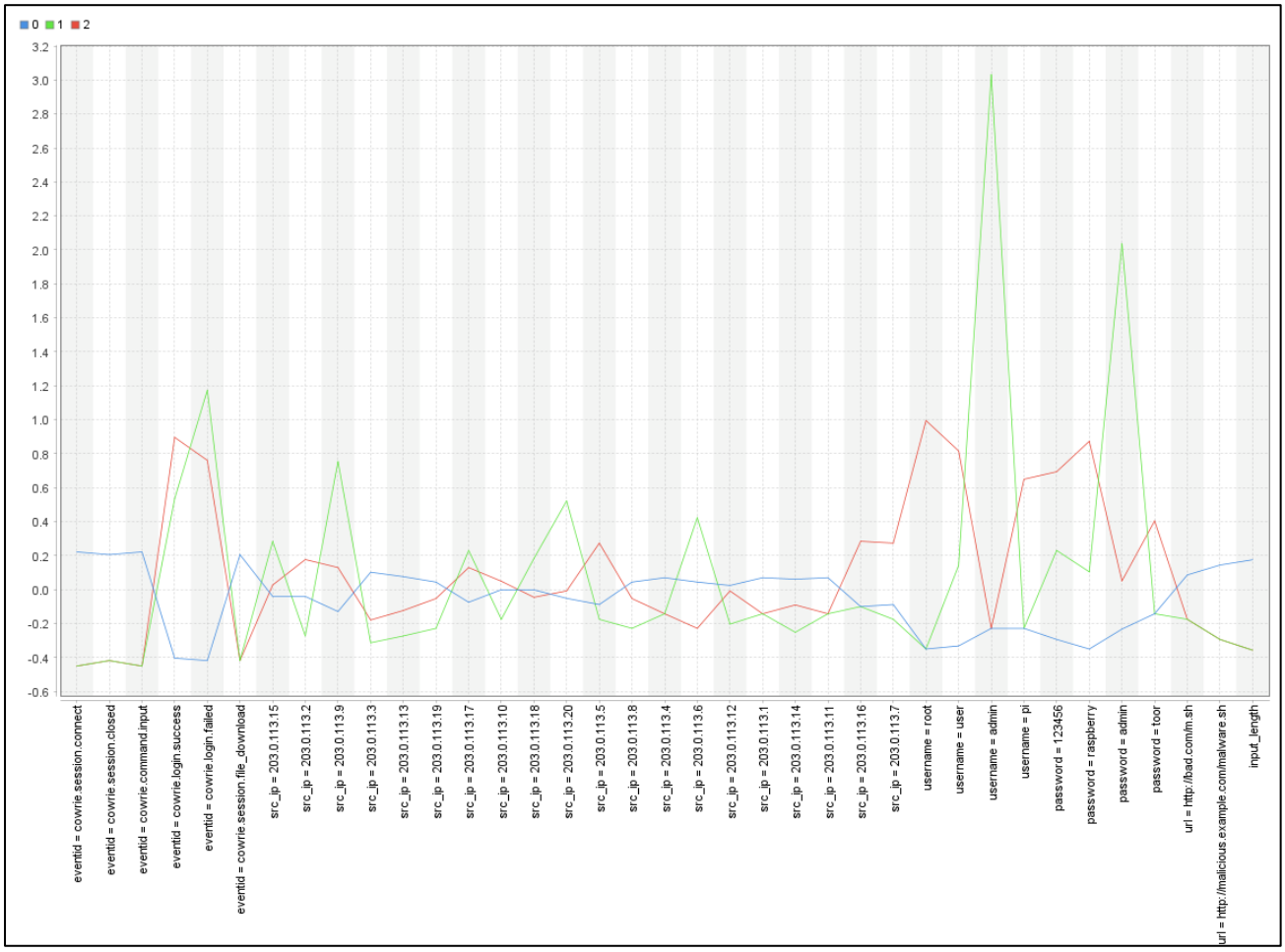


**Fig.3 RapidMiner Plot of K-Means Clustering**

**Data Preparation Steps:**
Retrieve Extended Data: Load your dataset.

Select Attributes: Choose only the relevant features for modeling.

**Generate Attributes:** Create new features if needed (e.g., derived columns).

**Replace Missing Values (twice):** Handle missing data by filling them with appropriate values (mean, median, mode, etc.).

**Nominal to Numerical:** Convert categorical data into numbers, as clustering algorithms require numeric input.

**Normalize:** Scale all features to the same range (usually 0–1) to ensure fair distance calculations during clustering.

**Select Attributes (2):** Final selection of attributes before clustering.

**Model Selection (Choosing the Clustering Algorithm):**

**Clustering Operator:**

This block is where the model is selected.

In clustering (unsupervised learning), the model could be K-Means. The selection depends on the nature of the data:

K-Means for globular clusters.

DBSCAN for arbitrary-shaped clusters and noise handling.

Training (Executing the Clustering Algorithm):

Once the data is cleaned, transformed, and normalized, it is fed into the clustering model. The model will then learn the patterns and group similar instances together based on feature similarity. It is a parallel coordinate plot (or line plot for clusters), a typical visualization after K-Means clustering.

Legend (Top Left):

0, 1, 2 → These are the cluster IDs (three clusters found).

Blue line = Cluster 0

Green line = Cluster 1

Red line = Cluster 2

**X-Axis:**

Different features/variables extracted from your Cowrie SSH honeypot logs.

e.g., eventid = cowrie.session.connect, eventid = cowrie.login.success, usernames, passwords, URLs tried, etc.

**Y-Axis:**

Normalized values for each feature (scaled between roughly -0.6 to 3.2). It shows the relative importance/intensity of each feature in a cluster. Each colored line shows the average pattern for that cluster across the features. Peaks and dips indicate which features are more dominant for that cluster.

For example:

Cluster 1 (Green) shows a very high spike for username = admin and password = admin → suggests attackers trying default admin credentials.

Cluster 2 (Red) spikes at username = root and password = 123456 → indicates a different attack behavior focused on root brute-forcing.

Cluster 0 (Blue) stays relatively flat → likely less suspicious or benign interactions.

URLs like http://bad.com/malware.sh being accessed are visible in the features; if these spike, it hints at attempts to download malware.

## 3.5 Integration and Automation

Following training and testing of the machine learning models with RapidMiner, the final clustering pipeline was installed as a background service on an Ubuntu server running the Cowrie SSH honeypot. Cowrie was set up to simulate a vulnerable SSH environment, drawing malicious login attempts and command executions from would-be attackers. Incoming SSH session logs, such as authentication attempts, command sequences, and connection metadata, were fed into the system continuously.

Preprocessing operations like data normalization and feature extraction were used to prepare the raw logs prior to passing them into the RapidMiner K-Means clustering model. The model partitioned the activities into separate clusters to facilitate differentiation between benign interactions, automated bot scans, and more complex intrusion attempts. This clustering analysis was done in close to real time, allowing for suspicious activity to be rapidly detected without the need for manual intervention.

Upon detecting abnormal clusters associated with potential threats, the system automatically generated alerts, notifying administrators of the suspicious activity.

The honeypot's function within the cybersecurity infrastructure underwent a dramatic change as a result of this integration. The improved Cowrie honeypot actively categorized, addressed, and manipulated attacker interactions in real-time, rather than just acting as a passive log collector for forensic analysis. The system was able to advance toward a more proactive, intelligent, and robust cyber defense strategy by combining the use of RapidMiner for clustering with reinforcement learning for adaptive deception.

# 4. RESULT AND DISCUSSION

## 4.1 Results

The K-Means clustering algorithm, applied through RapidMiner on the SSH honeypot logs collected from the Cowrie instance running on Ubuntu, successfully identified distinct patterns of malicious behavior. The model was configured to detect three clusters, each representing a different category of activity observed on the honeypot.

Taking K=3, the resulting parallel coordinates plot revealed clear behavioral differences among the clusters:

**Cluster 0 (blue line)** exhibited relatively flat and stable values across all features. This indicates a group of sessions with minimal engagement or random scanning behavior. Such traffic likely corresponds to benign or automated network scans without focused intrusion attempts.

**Cluster 1 (green line)** showed significant spikes for the features associated with the username admin and password admin, along with access attempts to known malicious URLs. This suggests a coordinated brute-force attack strategy leveraging common administrative credentials. Additionally, heightened values in command input length and file download events further indicate active exploitation attempts beyond simple login probing.

**Cluster 2 (red line)** displayed strong peaks for the root user and password combinations such as 123456 and raspberry. This pattern aligns with botnets or automated attack scripts targeting default credentials typically found on misconfigured or poorly secured systems.

The clustering results demonstrate that different attacker groups or bots exhibit distinguishable behavioral signatures when interacting with the honeypot. By grouping these interactions into clusters, the system is able to automatically classify and prioritize threat types without manual inspection. Notably, the system was able to differentiate between superficial scanning activity and deeper, more targeted intrusion efforts.

Furthermore, integrating these clustering results with the reinforcement learning (RL) agent allowed Cowrie to dynamically adapt its responses based on the detected threat profile. For example, more sophisticated deception tactics, such as fake vulnerabilities or staged file systems, could be selectively deployed against attackers showing signs of deeper engagement (Clusters 1 and 2), enhancing the system's effectiveness in intelligence gathering and threat deterrence.

Overall, the clustering approach provided valuable insights into attack behavior on the SSH honeypot, transforming raw log

data into actionable threat intelligence for proactive cyber defense.

## 4.2 Observations

The results demonstrate the effectiveness of K-Means clustering in profiling attacker behaviors based on SSH honeypot interaction logs. By categorizing sessions into three distinct clusters, it became possible to identify different threat patterns:

- general reconnaissance with minimal interaction,
- credential attacks using default administrative credentials, and
- root access brute-force attacks leveraging weak passwords.

The separation between clusters was particularly evident in key features such as username and password combinations, input length, and access to known malicious URLs. These findings confirm that even unsupervised learning methods can effectively reveal underlying attacker strategies without prior labeling of data.

Integrating this clustering model into the live honeypot system provides a strong foundation for real-time threat classification. Suspicious sessions can be rapidly flagged based on cluster membership, enabling timely alerts and adaptive response mechanisms. Furthermore, coupling the clustering engine with a reinforcement learning agent enhances the honeypot's ability to dynamically deceive attackers, tailoring fake system behaviors to specific attacker profiles.

Thus, the combined machine learning pipeline transitions the honeypot from passive logging to proactive engagement, significantly strengthening cyber defense capabilities.

# 5. CONCLUSION AND FUTURE SCOPE

This research successfully demonstrated the significant benefits of integrating machine learning techniques with honeypot systems to enhance cybersecurity defenses. Through a well-structured methodology encompassing data collection, feature engineering, model training, and real-time deployment, the project established a comprehensive framework capable of intelligent threat detection, classification, and response. The results highlighted that machine learning not only improves the accuracy and speed of detecting cyber threats but also introduces adaptability and predictive capabilities, which are absent in traditional honeypot implementations. This combination of deception and intelligence forms a proactive line of defense, capable of countering both known and emerging threats with increased efficacy.

The study confirms several hypotheses. First, ML-enhanced honeypots demonstrate significantly higher accuracy in detecting and classifying malicious activity compared to static, rule-based systems. Second, unsupervised and reinforcement learning models show particular promise in identifying novel and evasive threats, contributing to a more robust security posture. Third, by analyzing behavioral trends and attacker interaction patterns, the system can predict future attacks and take preemptive measures. Moreover, despite the computational complexity of certain ML models, careful optimization ensured that real-time detection was achieved without substantial performance overhead. Overall, the system represents a transformative step toward automated and adaptive cyber defense.

Looking forward, there are several promising directions for future research. Deep learning models, especially those based on recurrent neural networks (RNNs) and transformers, can be explored to analyze time-series data and sequential command patterns for even more accurate detection. Deployment of the system on edge devices and integration with distributed computing platforms can further enhance its scalability and resilience. Additionally, incorporating federated learning will enable the training of models across multiple honeypot instances while preserving data privacy. Integration with Security Information and Event Management (SIEM) tools can enable enterprise-wide alert correlation and automated incident response. Finally, future work should focus on strengthening the system's robustness against adversarial ML attacks and implementing concept drift detection to ensure long-term adaptability.

In conclusion, the fusion of honeypot deception techniques with machine learning intelligence marks a paradigm shift in cybersecurity. This hybrid approach not only enriches threat detection and analysis but also lays the groundwork for developing resilient, adaptive, and self-learning security systems capable of withstanding the evolving landscape of cyber threats.

This research demonstrates that integrating machine learning with honeypot systems significantly enhances the detection, classification, and mitigation of cyber threats.

The system:

- Collects rich interaction data.
- Uses ML for intelligent classification.
- Employs adaptive strategies for dynamic threat engagement.
- Supervised models are effective for known threats, while unsupervised models uncover novel attacks. Reinforcement learning brings adaptability to a traditionally static security tool.

**Future Scope**

- Advanced Deep Learning Models: Incorporate LSTM or Transformer-based models for detecting temporal attack patterns.
- Edge Deployment: Optimize models for edge devices in distributed network environments.
- Federated Learning: Train across multiple honeypot nodes without centralized data sharing, ensuring privacy.
- Integration with SIEM Systems: Seamless correlation with enterprise-wide event management platforms.
- Adversarial Defense: Enhance ML models against adversarial inputs and evasion techniques.

# 6. ACKNOWLEDGMENTS

I would like to express my gratitude to Prof. L. Hemochandra Singh, Professor of Statistics in the Department of Basic Science at the College of Agriculture in Imphal, and to Dr. Maibam Romio Singh for their unwavering support in all areas.

# 7. REFERENCES

[1] Shyamalendu Paul, Amitava Podder, Kaustav Roy, (2024), Exploring the Impact of AI-based Honeypots on Network Security, Educational Administration: Theory and Practice, 30(6), 251-258, Doi: 10.53555/kuey.v30i6.5155

[2] Iyer, Kumrashan Indranil. (2021). Adaptive honeypots: Dynamic deception tactics in modern cyber defense. International Journal of Science and Research Archive. 04. 340-351. 10.30574/ijsra.2021.4.1.0210.

[3] Dakic, Vedran & Regvart, Damir. (2025). Advancing

Cybersecurity with Honeypots and Deception Strategies. Informatics. 12. 14. 10.3390/informatics12010014.

[4] Narayana Gaddam. (2025). AI-enhanced honeypots for advanced cyber deception strategies. QIT Press - International Journal of Cyber Security Research and Development (QITP-IJCSRD), 5(1), 9–19.

[5] J. Franco, A. Aris, B. Canberk and A. S. Uluagac, "A Survey of Honeypots and Honeynets for Internet of Things, Industrial Internet of Things, and Cyber-Physical Systems," in IEEE Communications Surveys & Tutorials, vol. 23, no. 4, pp. 2351-2383

[6] Martínez S., C. J. ., Moreno A., H. O. ., & Hernández A., M. B. . (2023). Analysis of Intrusions into Computer Systems using Honeypots. International Journal of Intelligent Systems and Applications in Engineering, 11(6s), 461–472. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/2871

[7] Sokol, P., Míšek, J. & Husák, M. Honeypots and honeynets: issues of privacy. *EURASIP J. on Info. Security* **2017**, 4 (2017). https://doi.org/10.1186/s13635-017-0057-4

[8] Mokube, Iyatiti & Adams, Michele. (2007). Honeypots: concepts, approaches, and challenges. 321-326. 10.1145/1233341.1233399.

[9] Bharadiya, Jasmin. (2023). Machine Learning in Cybersecurity: Techniques and Challenges. European Journal of Technology. 7. 10.47672/ejt.1486.

[10] V. -I. Năstase, M. -E. Mihăilescu, S. Weisz, L. V. Dagilis, D. Mihai and M. Carabas, "Cowrie SSH Honeypot: Architecture, Improvements and Data Visualization," 2024 23rd RoEduNet Conference: Networking in Education and Research (RoEduNet), Bucharest, Romania, 2024, pp. 1-7, doi: 10.1109/RoEduNet64292.2024.10722609

[11] Krajčík, Patrik & Mikuláš, Matúš & Helebrandt, Pavol & Kotuliak, Ivan. (2025). Improvement of Cowrie honeypot interaction and deception capabilities. 1-9. 10.1109/KIT67756.2025.11205433.

[12] Li, Youguo & Wu, Haiyan. (2012). A Clustering Method Based on K-Means Algorithm. Physics Procedia. 25. 1104-1109. 10.1016/j.phpro.2012.03.206.

[13] Zhang, Chaoyu & Wang, Ning & Hou, Y & Lou, Wenjing. (2025). Machine Learning-Based Intrusion Detection Systems: Capabilities, Methodologies, and Open Research Challenges. 10.36227/techrxiv.173627464.48290242/v1.