JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Real-Time Honeypot with Behavioral Fingerprinting of Attackers Using an ML Model

[1]Anubhuti Jain, [2]Laukic Aiyappa, [3]Meghana M J, [4]Umme Aamina A, [5]Dr. Mohammed Tajuddin

[1234]Student, [5]Professor & HoD

[12345]Computer Science Engineering (Cyber Security)

[12345]Dayananda Sagar College of Engineering, Bangalore, India

Abstract : This project paper gives the design of our real-time honeypot system.The proposed honeypot system captures the SSH/telnet and the web based attacks.This honeypot forwards all the collected data to a cloud database. In the analysis kayer it uses a machine learning model to fingerprint the attackers based on the logs collected. SSH/Telnet logs are collected through Cowrie and Web based exploits are collected by Glastopf. The logs are sent in real-time to supabase. Rule based heuristics will handle the brute force or the sql injection payloads. The ML classifiers like SVM, Random forest, or XG boost analyze the collected logs and categorize the attackers based on their behavior. The outputs are visualized using the Grefana dashboards which shows connection counts , geolocations, login password patterns etc. The alerts are triggered for anomalous patterns.

IndexTerms **- Honeypots. Deception. Attacker fingerprinting. Machine learning. Cowrie. Glastopf. Supabase. Grafana. Cyber security**

I. INTRODUCTION

Honeypots are the decoy systems which are deceptive in nature. They're designed to lure attackers and reveal their tactics [11], [16], [24].By simulating vulnerable services,these honeypots observe the attack patterns like repeated login attempts , exploit payloads etc. in controlled environment.Such intelligence can reveal an attackers origin. Tools. Objectives. It guides stronger defenses. For example. Monitoring a Cowrie Bangalore, India tajdsce@gmail.com SSH Telnet honeypot will log brute force credentials. Com mands. A Glastopf web honeypot logs malicious URLs. Pay loads. This information is invaluable for user attacker behavior analytics in UBA [16]. Gartner defines that as profiling users. Detecting anomalies. Krasznay and Hamornik specifically advocate UBA to expose targeted attacks. Those unseen by traditional defenses.

Modern research has also explored integrating honeypots with machine learning. Threat intelligence. HoneyMustard is an adaptive honeypot framework [1]. It emulates realistic user behavior in real time. That improves deception fidelity. From reference one. Similarly. Gandara et al compare multiple hon eypot systems. Including Cowrie and Glastopf [11]. They note emerging trends toward ML driven and cloud based honeypots. Other works use ML to configure honeypots dynamically. For example. Clustering Nmap scan results to generate custom configurations. From twenty. Twenty five. Twenty six. In parallel. Analysts have used attack pattern classification to enhance security [3], [4], [19]. Seid et al analyzed two hundred f ifty four incident reports. They identified common multi stage attack patterns. Mapped them to CAPEC signatures. These patterns inform our approach to fingerprint attacker behavior. Weextend these ideas by integrating Cowrie and Glastopf logs into a unified pipeline. We apply heuristics. ML to fingerprint attackers in real time. We visualize the results.

This paper outlines our Phase 1 design. Methodology for a Real Time Honeypot with Behavioral Fingerprinting. We review related work in Section 2. We describe the proposed system architecture. Data flow in Section 3. We detail the implementation environment setup in Section 4. We discuss preliminary conceptual results in Section 5. Phase 1 culminates in a designed pipeline. It is ready for deployment. ML model integration.

II. LITERATURE REVIEW

Recent research highlights the value of honeypots. Behav ioral analysis in cybersecurity [6], [7], [17]. Liu et al introduce HoneyMustard [1]. That is a framework that performs real time. Application level user behavior emulation. It makes honeypots more realistic. Harder to detect. Similarly. Moric et al conduct a comprehensive comparative study [11]. Including Cowrie and Glastopf. They assess their capabilities. They advocate integration of ML. Cloud technologies for adaptive detection. Zhang and Shi propose a dynamic honeypot. It uses ML [20], [25]. Nmap scans of network devices are clustered. To auto generate honeypot configurations. It actively adjusts to the network environment.Various researchers have categorized honeypots. Based on their level of interaction. Purpose. Table I summarizes the main types. Examples. Use cases. Drawn from prior comparative studies.

Table 1 Comparison of Honeypot Types

Type	Examples	Typical Use Case
Low-Interaction	Glastopf, Dionaea	Web probes, quick malware capture; low maintenance and low risk.
Medium-Interaction	Cowrie, Kippo	Capture SSH/Telnet credentials and commands for behavioral analysis.
High-Interaction	Argos, Honeywall	Full attacker engagement for deep behavior profiling; higher risk and resource use.
Client Honeypot	Thug, Honeyclient	Act as a client to detect drive-by exploits and malicious web content.

In attacker behavior analysis, Krasznay and H´ amornik point out that profiling users and spotting anomalies through UBA helps uncover those quiet targeted attacks [3], [4]. Seid and his team took that idea and used it for critical service providers.They studied the actual incident reports. After study ing they picked out give attack scenarios and found seven vulnerabilities. Along with that they also identified ten attack patterns and to analyse their behaviours they referred to the CAPEC and MITRE frameworks data. This give a very clear and structured view into the attacks that are carried out in multiple stages. We referred this paper because we want to pull out similar kind of results in the behavioural fingerprinting.

Researchers have started using machine learning in the honeypot systems . Some of the researchers came up with HoneyModels which are machine learning models incorpo rated with honeypots. They have put a hidden watermark or trapdoor inside the model. This system can detect the attacks and has accuracy of 69.5 percent. Hence it shows that the deception and the trap techniques work effectively. In another study we saw that a group added trapdoor examples in the neural networks . This will lead to attackers to create inputs with similar kinds of hidden features accidentally. Then the honeypot can easily detect those and other malicious behaviour [23], [30].

However honeypots have this one big problem which is that the attackers can often identify and fingerprint them [19], [29]. Srinivasa and his team had created multistage system to fingerprint the honeypots. They scanned around 2.9 billion ipv4 addresses and found around 21,000 honeypots. This tells us that the attackers can easily find and avoid these types of honeypot systems. Their research also provides different ways in which we can make our honeypot harder to detect. Referring to their study we get to know the importance of improving honeypots over time. Overall the researches referred provide many insights for our project where we deploy the honeypots and collect the data , use behavioural fingerprinting to profile the attackers and outputs are shown in dashboards.

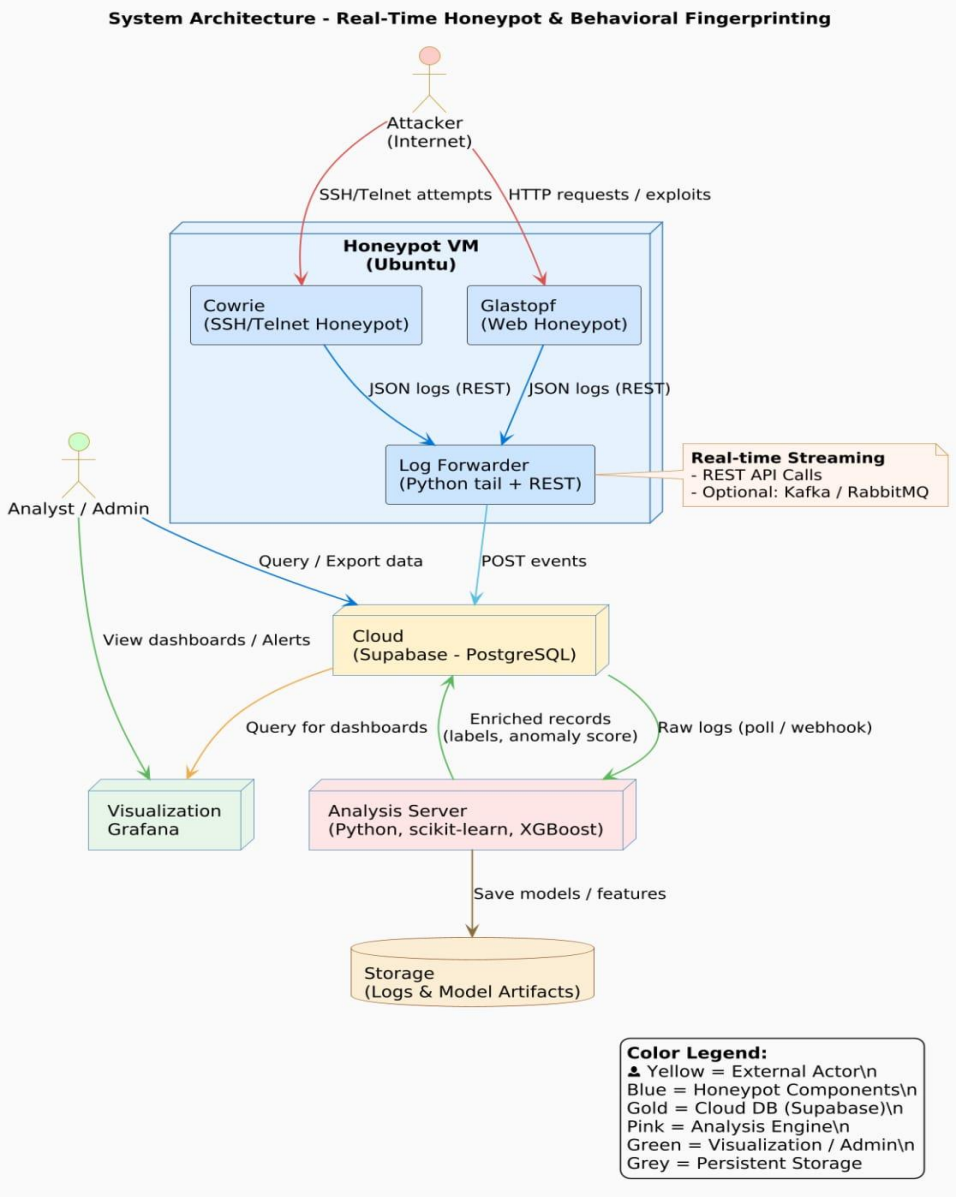
III. METHODOLOGY AND PROPOSED SYSTEM

Our system has two main honeypots which are a. Cowrie for ssh/telnet and b. Glastopf for web based attacks. Both of these send the data in real time for the analysis as shown in f igure 1.Cowrie pretends to be a real ssh shell and records all of the login attempts made, password guesses and the commands executed by the attacker [6]. Glastopf is used for web based attacks and it pretends to have web vulnerabilities like prone to sql injections,css etc. It records and logs all of the malicious http requests [5], [6]. Both of the honeypots run on the Ubuntu VM . Their ports are exposed to the internet. Each of the honeypots log attempts like the connection attempts , username or passwords tried, commands that are entered,and attack payloads. The logs collected are immediately sent to the supabase cloud PostgreSQL database using the REST api calls.

Before these logs are stored, a rule based layer checks and analyses each event for example for many many of the fast failed ssh login attempts from the same IP address , it marks as the the brute force attacks. These rules will create the labels for the attackers. At the same time we also extract the features from the data like number of events, session duration, commands or urls, user agent values etc. These features are sent into machine learning classifiers which are built using the scikit learn . These classifiers are designed to predict the type of the attacks such as brute force ssh, we scanning , malware dropper etc [9], [21], [23]. Each session will get its own behavioural fingerprint and those results are stored back in the database.

Grefana dashboard connects to the supabase and refreshes the dashboards regularly. The dashboard shows time series charts or attack counts , maps of attackers IP locations, histograms of usernames/ passwords tried etc. We also set the alerts in case there is a sudden spike in the attacks, the analyst gets notified. This real time behaviour helps respond quickly. [2], [15], [25].

Fig. 1: Architecture of the proposed honeypot system



IV. IMPLEMENTATION DESIGN(ENVIRONMENT SETUP AND PIPELINE)

The plan of phase one in this implementation is laid out. It includes the process of data collection and processing pipeline setup.

In the case of the virtual machine, we configured an Ubuntu 22.04 LTS server that has a single CPU, two gigabytes of RAM, and twenty gigabytes of disk. Firewall rules and a public IP address are assigned to this VM. Those rules allow SSH on port twenty-two to connect the real side and SSH on port twenty-two twenty-two to connect the honeypot side. On port eighty, comes HTTP.

With regard to the deployment of the honeypot, we install Cowrie and Glastopf directly on the VM using their official source. Cowrie is used as a shell in Linux mimicking the shell prompt, and making faked files instead of real files [6], [24]. Glastopf manages such typical web triggers like SQL injection and file inclusion problems. Each of the two tools records all activities in CSV or JSON.

Data forwarding occurs via a primitive python service that we develop. It leverages the requests library to make the honeypot logs and send every occurrence as a POST to the Supabase REST endpoint immediately. We develop a Supabase application based on PostgreSQL. The SSH logs and the HTTP logs are contained in tables within it. Each record includes a time stamp, an origin IP address, a port or a protocol that was utilized, request information such as commands or URLs, and the actual content of the request itself [2].

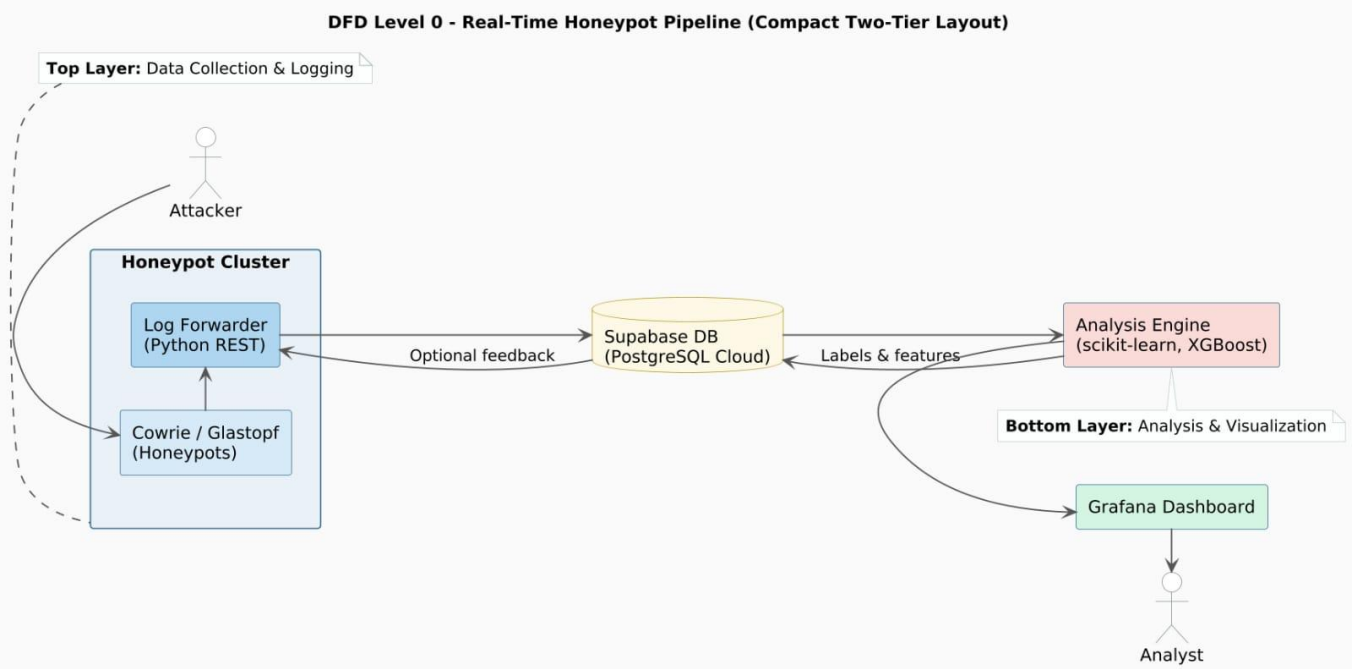
The server where the analysis is done is another machine which may be either Linux or windows server. We set up Python three point nine in that location, together with pandas, scikit-learn, and some others. The Supabase analysis Python code verifies new records at intervals. It extracts such char acteristics as the number of attempts per minute, percent ages of successful logins, and tendencies in using particular commands. Then it executes pre-trained machine learning on them. They are SVM, random forest and XGBoost models. We initially train them with open dataset historical labeled data or previously acquired honeypot runs. They store as pickle files to be used later. Finally, the analysis will insert classification labels and anomaly scores into the database.

Table 2 Key Components of Proposed Honeypot System

Component	Technology Used	Purpose
SSH/Telnet Honeypot	Cowrie on Ubuntu VM	Capture login attempts, record commands and session metadata.
Web Honeypot	Glastopf	Emulate vulnerable endpoints, log exploit payloads and malicious URIs.
Database	Supabase (PostgreSQL)	Store raw logs and parsed features for real-time analysis.
ML Models	SVM, Random Forest, XGBoost	Session classification, anomaly scoring, and fingerprint generation.
Visualization	Grafana	Dashboard panels (time-series, geolocation, credential scatter) and alerting.

In the case of Grafana configuration, we mount it on windows eleven or Linux. Next in sequence comes the Post greSQL plug in and we point the data source to Supabase. Dashboards are imported or made manually. They can be pan els with SSH login attempts history, top countries by attackers, failed and successful logins, the most frequent payloads. We also installed the Grafana Alerts by creating alert rules. By way, a warning email will be issued when an IP makes more than fifty unsuccessful attempts to log in per minute.

Fig. 2: Flow diagram of the proposed honeypot system



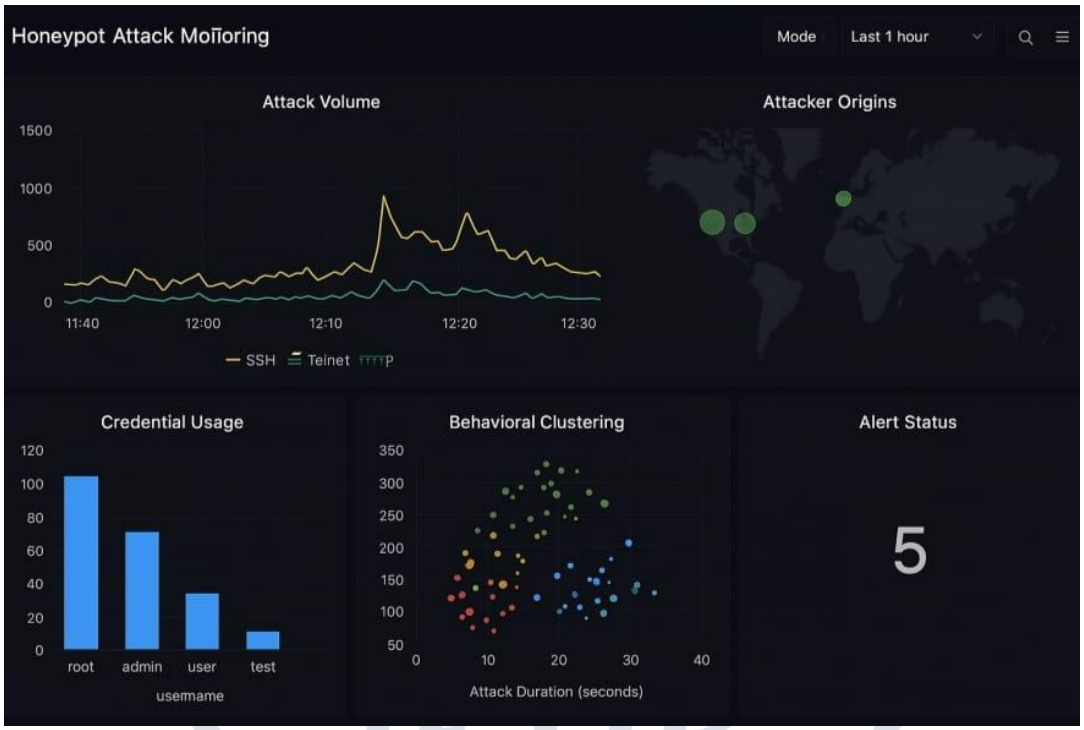
This is the overall flow of the pipeline. The Supabase database takes feeds of honeypots. Python analysis withdraws there and puts results back. It all can be seen in Grafana dashboards. We will use containers using Docker wherever it is necessary, simply to ensure things can be reproduced. At the conclusion of this phase, the surrounding and the flow of data are prepared. Phase two entails detailed machine learning tuning and the entire dataset collection.

V. EXPECTED OUTCOMES

The results to be expected are a real-time modular honeypot infrastructure. It collects and categorizes attacker's activity between the protocols. After coming up and running, the hybrid configuration where Cowrie is used to support SSH and Telnet as well as the Glastopf to support HTTP continue to generate structured logs. Those records are sent directly to a PostgreSQL database on Supabase that is hosted on clouds.

The machine learning pipeline incorporates other classifiers such as support vector machine, random forest as well as XGBoost. It iscribing a distinct behavior representation of the fingerprints of the attack session. Such fingerprints follow things like frequency of logins, command entropy, variety of the payload and delays between actions. This kind of an arrangement can distinguish between the brute-force bots and the reconnaissance tools or even the focus attacks.

Fig. 3: Mock-up of the expected dashboard



These results will be represented in a Grafana dashboard in the form of dynamic charts and alert panels. The important visual elements that are anticipated to be included are:

- Real-time attack feed: time-series plots of the attack's strength by protocol.
- Geolocation maps: area of source of attackers.
- Credential analysis panel: most used usernames/ pass words.
- Behavioral clustering: Clustering of attackers based on similarity scores of ML.

Figure 3 provides a conceptual mock-up of what people would want to see of the dashboard interface. It highlights the time- series graphs and the overlays of geographic features and all warnings.
Such visualization has a couple of benefits to incident response teams. It gives a preliminary clue on behaviors. It also helps in matching repetitive patterns of attack. And it also tells ingrained defense that can be altered accordingly.

VI. CONCLUSION AND FUTURE SCOPE

The present Phase 1 research had assembled a real time system that utilized honeypots to perform behavioral finger printing of attackers. In the configuration, we combined two honeypots which are complementary to each other. One of them is Cowrie (SSH, Telnet). The other one is the Glastopf web interactions. Their records are sent to a Supabase cloud database in real time. We have described the pipeline involving the heuristic rule-based and ML classifiers. These are SVM, Random Forest and XGBoost. The aim is to tag attacker ses sions based on their behavioural pattern. Grafana dashboards are going to provide real-time attack patterns. They are also useful in alerting anomalies, as well as supporting alerting on anomalies [8], [18], [23]. We have reviewed the literature on previous research on adaptive honeypots. It also considered the attack pattern analysis. And it involved ML-based deception methods. That was all we used to design by. The initial mock ups show that the system is able to reveal major insights of honeypot data. Such issues as attempts to use logins and the use of commands become evident. At end of phase 1, the environment and data pipelines are in place. In subsequent stages we will gather real life data. The ML models will be trained on attack logs that are labeled. And we will judge the performance of classification. That ought to accomplish the real time behavioral fingerprinting system we have fancied.

The implementation will be further expanded to cover in the following stage:

- Training of full ML model with gathered attack data and validation on a variety of datasets.
- Combination of rule based heuristics and probabilistic modeling to make hybrid decisions.
- Grafana triggered Telegram or Slack alerting using bots.
- Real attack traffic performance assessment and accuracy, recall and precision measures calculation.

The system that was completed must be a useful research platform. It is centered on dynamic threat intelligence. And it assists in proactive network defense in business and university settings.

VII. REFERENCES

[1] S. Liu, S. Wang and K. Sun, "Enhancing Honeypot Fidelity with Real-Time User Behavior Emulation," IEEE Access, 2023.

[2] D. R. Rodríguez, S. Romeu, I. Gimenez and T. R. Catala, "Enhancing Cybersecurity Intelligence through Machine Learning: Clustering and Forecasting Analysis of Honeypot Data," Cybersecurity Agency of Catalonia, 2024.

[3] K. Majumdar, N. Kumar, A. Handa and S. K. Shukla, "Attackers' Profiling Based on Multi-Attack Patterns in SSH Service," arXiv preprint, 2025.

[4] V. Chintamaneni, M. SreeRamu, P. Nagarjuna, P. Meekala and P. Sumegha, "Unmasking Cyber Adversaries: Leveraging Cyber Threat Intelligence for Attacker Behavior Analysis," International Journal of Cybersecurity Research, 2024.

[5] M. Abewa and T. Melese, "Dynamic Interactive Honeypot for Web Application Security," International Conference on Web Security, 2024.

[6] A. Boparai, R. Ruhl and D. Lindskog, "The Behavioural Study of Low Interaction Honeypots: DShield and Glastopf in Various Web Attacks," Journal of Cybersecurity Studies, 2024.

[7] K. N. Mallikarjunan, S. Prabavathy, K. Sundarakantham and S. M. Shalinie, "Model for Cyber Attacker Behavioral Analysis," Procedia Computer Science, 2015.

[8] Y. Yang, S. Sun and W. Wang, "ShellBox: Adversarially Enhanced LLM-Interactive Honeypot Framework," IEEE Access, 2025.

[9] C. Guan, G. Cao and S. Zhu, "HoneyLLM: Enabling Shell Honeypots with Large Language Models," IEEE Transactions on Information Forensics and Security, 2023.

[10] A. Vasilatos, N. Papadopoulos, L. Petrou and E. Vassilakis, "LLMPot: Dynamically Configured LLM-based Honeypot for Industrial Protocol and Physical Process Emulation," IEEE Access, 2025.

[11] D. Moric, M. Dakic and J. Regvart, "Advancing Cybersecurity with Honeypots and Deception Strategies," Journal of Information Security Research, 2025.

[12] Y. Zou, F. Chen, P. Li and H. Zhang, "Developing High-Interaction Honeypots to Capture and Analyze Region-Specific Bot Behaviors," IEEE Security and Privacy Workshops, 2024.

[13] M. Odemis, C. Yucel and A. Koltuksuz, "Detecting User Behavior in Cyber Threat Intelligence: Development of Honeypot System," Journal of Information Security and Applications, 2022.

[14] L. Constantino, "Quantifying the Effectiveness of Dynamic Response in Web Application Honeypots," IEEE Access, 2025.

[15] S. K. S. H. and C. Saravanan, "A Comprehensive Study on Data Visualization: Grafana," International Journal of Emerging Technology Research, 2021.

[16] C. Krasznay and B. Hamornik, "Analysis of Cyberattack Patterns by User Behavior Analytics," Computers & Security, 2018.

[17] A. Kubba, T. Ivanov and R. Bell, "A Systematic Review of Honeypot Data Collection, Threat Intelligence Platforms, and AI/ML Techniques," ACM Computing Surveys, 2025.

[18] S. Gaddam, "AI-Enhanced Honeypots for Advanced Cyber Deception Strategies," IJNRD, 2025.

[19] A. Sayed, R. Gupta and M. Al-Zahrani, "Honeypot Allocation for Cyber Deception in Dynamic Tactical Networks: A Game-Theoretic Approach," IEEE Transactions on Network Science and Engineering, 2023.

[20] J. Lin, P. Das and R. Wu, "Optimizing Internet of Things Honeypots with Machine Learning," Journal of IoT Security, 2024.

[21] S. Lanka, P. Varma and K. Menon, "Intelligent Threat Detection — AI-Driven Analysis of Honeypot Data to Counter Cyber Threats," IEEE Access, 2024.

[22] P. Aggarwal, S. Mehta and R. Khanna, "Decoys in Cybersecurity: An Exploratory Study to Test the Effectiveness of Two-Sided Deception," Computers & Security, 2021.

[23] E. Ebunoluwa, T. Johnson and M. Aruna, "AI-Powered Honeypots: Enhancing Deception Technologies for Cyber Defense," IEEE Transactions on Information Forensics and Security, 2025.

[24] M. Franco, L. Gonzalez and P. Romero, "A Survey of Honeypots and Honeynets for IoT, IIoT and Cyber-Physical Systems," IEEE Internet of Things Journal, 2021.

[25] A. Taylor, V. Brown and S. Patel, "Machine Learning-Driven Honeypots in Zero Trust Networks," IEEE Access, 2025.

[26] R. Sharma and M. Gupta, "AI-Driven Adaptive Honeypots for Dynamic Cyber Threats," IJNRD, 2024.

[27] M. Ahmed, "Securing Smart Cities through Machine Learning: A Honeypot-Driven Approach to Attack," IJNRD, 2024.

[28] V. Morozov, N. Mikhailov and D. Orlov, "An Adaptive Honeypot Framework for Design and Evaluation," IEEE Internet of Things Journal, 2024.

[29] T. Evans, P. Lee and S. Park, "Analysis of Methods of Attracting Attackers in the Honeypot," Cybersecurity Journal, 2025.

[30] G. Sladic, M. Popovic and T. Zivkovic, "VelLMes: A High-Interaction AI-Based Deception Framework," IEEE Access, 2025.

[31] V. S. Deepthi and V. S. Vagdevi, "Behaviour Analysis and Detection of Blackhole Attacker Node under Reactive Routing Protocol in MANETs," International Conference on Networking, Embedded and Wireless Systems, 2018.