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

SSH-HONEYGUARD: AN EXPLAINABLE AND CONTAINER-ISOLATED SSH HONEYPOT FOR ATTACK BEHAVIOR ANALYSIS

Capturing, Classifying, and Correlating Attacker Behavior through Isolated Deception and Rule-Based Detection

[1]Pravin Kounder, [2]Sehar Khan
[1]MSc CS specialization in cybersecurity, [2]Assistant Professor
[1,2]Department of Advanced Computing
[1,2]Nagindas Khandwala College
[1,2]University of Mumbai, Maharashtra, Mumbai

Abstract: SSH services are frequently targeted by automated and manual attacks, making them useful sources of information for studying attacker behavior. Conventiona SSH honeypots can capture authentication attempts and commands, but the collected information often requires manual analysis to understand the progression and severity of an attack. Existing research has also explored container-based deception, adaptive honeypots, attacker profiling, and automated analysis, but practical systems still face limitations related to deployment complexity, analysis speed, or integration of multiple security functions [1], [10]. This paper presents **SSH-HoneyGuard**, a container-isolated SSH honeypot designed to capture and analyze attacker interactions using an explainable rule-based detection approach. The system uses a Paramiko-based SSH service, a controlled fake shell, structured session and authentication logging, command categorization, and detection of suspicious and multi-stage activities. Docker isolation is used to separate the honeypot environment from the host system. SQLite provides persistent storage for sessions, authentication attempts, commands, and security alerts. The implemented system was evaluated through a series of controlled experiments covering command classification, multi-stage attack detection, authentication logging, container isolation, and persistence of collected data. The final dataset contained 52 sessions, 50 authentication attempts, 409 recorded commands, and 147 generated alerts. Of the recorded commands, 328 were classified into defined behavioral categories, corresponding to approximately 80.20% classification coverage. The system also generated 11 multi-stage activity alerts, demonstrating its ability to correlate suspicious activities across attack stages.

Keywords: SSH honeypot, cyber deception, Docker isolation, attack detection, behavioral analysis, command classification, multi-stage attack, cybersecurity.

1. INTRODUCTION

Secure Shell (SSH) is widely used for remote administration and therefore remains an attractive target for attackers. SSH honeypots provide a controlled environment in which authentication attempts, sessions, and attacker commands can be observed without exposing production systems. Previous honeypot research has explored different forms of deception, distributed deployment, adaptive interaction, and attacker behavior analysis [1], [4]. Container-based honeypots provide an additional security boundary by separating the deceptive environment from the underlying host. HoneyFactory demonstrated that container-based cyber deception can provide scalable honeynet deployment and deeper attacker interaction [1]. Other studies have investigated cloud-based honeypots and containerized deception to improve the collection of attack information [2], [3]. Research has also examined adaptive interaction and attacker

profiling as ways to improve the usefulness of honeypot data [5], [9]. However, collecting attacker activity alone does not necessarily provide an immediate understanding of what the attacker is attempting to achieve. Raw commands and authentication records still require analysis before they can be converted into meaningful security information. Automated approaches can reduce this burden by categorizing commands and identifying behavioral patterns. The main objective of SSH-HoneyGuard is therefore to combine **SSH interaction capture, container isolation, command classification, and explainable detection** within a lightweight research system. Instead of executing attacker commands on the real host, the system provides a controlled fake shell and records submitted commands as data.

The main contributions of this work are:

1. A Docker-isolated SSH honeypot architecture for controlled attacker interaction.
2. Structured collection of authentication, session, and command activity.
3. Rule-based categorization of suspicious SSH commands.
4. Detection of multi-stage attacker behavior.
5. Experimental evaluation using controlled attack scenarios and actual collected data.

Existing SSH honeypot approaches demonstrate the value of observing attacker interactions, but there remains a practical need for lightweight systems that combine interaction logging, behavioral classification, multi-stage activity detection, and isolated deployment within a single architecture. This work addresses that need through SSH-HoneyGuard, a containerized SSH honeypot designed to record attacker behavior without executing received commands. The system further analyzes command sequences to identify suspicious activities and correlate multiple attack stages.

2. RELATED WORK

Honeypot research has developed from simple deceptive services toward more interactive and analytically capable cyber-deception environments. HoneyFactory introduced a container-based comprehensive honeynet architecture that supports multiple honeypots and deeper attack observation [1]. Its results demonstrate the usefulness of container-based environments for collecting detailed attack information. H-DOCTOR investigated the use of honeypot information for firewall tuning, demonstrating how captured attack information can support defensive security decisions [2]. Kelly et al. compared honeypots across different cloud platforms and highlighted the importance of deployment environments when evaluating honeypot performance [3]. Lanka et al. investigated AI-driven analysis of honeypot data for threat detection [4]. Such approaches demonstrate the potential of automated analysis, although computational requirements and processing time can influence practical deployment. Adaptive approaches have also been explored. Huang and Zhu proposed reinforcement-learning-based adaptive honeypot engagement, where interaction strategies can be adjusted according to attacker behavior [5]. Möller proposed an adaptive multi-layered honeynet architecture that uses deep-learning-based anomaly detection and a reinforcement-learning agent to escalate sessions from low-interaction to high-interaction honeypots [6]. These studies indicate a movement toward dynamic environments capable of responding to attacker activity. Large-scale datasets are another important research direction. The MURHCAD work provides a multi-regional cloud honeypot dataset that can support research into geographically distributed attacker activity [7]. Zambianco et al. investigated MITRE ATT&CK-based decoy selection for cyber deception [8], while Valeros et al. examined attacker profiling through geographically distributed honeypots [9]. Containerized deception has also been investigated as a practical method for tracking attackers while maintaining separation from production infrastructure [10]. Despite these developments, the reviewed work shows that honeypot systems often emphasize one or two aspects such as deception, adaptive interaction, data collection, or automated analysis. SSH-HoneyGuard focuses on bringing several lightweight capabilities together, particularly **isolated SSH interaction, structured logging, explainable command classification, and multi-stage activity detection**.

3. METHODOLOGY

The methodology of SSH-HoneyGuard is based on controlled SSH attack observation. The system was developed incrementally so that the SSH interaction layer, isolation layer, logging components, and detection components could be tested independently before being combined. The experimental environment consists of a Windows host using WSL2 Ubuntu, Docker, Docker Compose, Python, Paramiko, OpenSSH, and SQLite. The project is maintained under the ssh-honeyguard directory, while the honeypot itself operates inside a Docker container. The development environment follows the architecture documented for the project. The SSH service accepts connections through the Paramiko implementation. Each connection is associated with a unique session identifier and corresponding session information. Authentication attempts are recorded with the username, authentication method, result, source address, and timestamp. After authentication, the attacker receives a controlled fake SSH shell. Commands entered by the attacker are treated strictly as input data. The fake shell does not pass

attacker-controlled strings to subprocess, os.system, eval, exec, or an actual operating-system shell. Instead, known commands receive predefined responses and unknown commands receive a harmless command-not-found response.

Submitted commands are classified into behavioral categories including:

- Reconnaissance
- Credential access
- Privilege escalation
- Persistence
- Network activity
- Download activity
- Destructive activity
- File activity
- Unclassified activity

The detection layer evaluates classified commands and generates security alerts for suspicious behavior. When activity crosses multiple behavioral stages, the system can generate a **multi-stage activity alert**.

The experiments were conducted as controlled demonstrations against the locally deployed honeypot. The final dataset was then queried from the SQLite database to obtain quantitative measurements.

4.SYSTEMDESIGN AND IMPLEMENTATION

SSH-HoneyGuard follows a modular architecture consisting of the SSH service, authentication logger, fake shell, command logger, classification logic, detection engine, and persistent storage.

4.1 SSH Interaction Layer

The SSH server is implemented using Paramiko. It accepts SSH connections and establishes a separate session for each connection. The session component records information such as session identifier, source address, start time, end time, and session outcome.

4.2 Authentication Logging

Authentication activity is stored independently from command activity. Each authentication record contains the session identifier, source IP, source port, username, authentication method, result, and timestamp. This separation allows authentication behavior to be analyzed independently from later commands.

4.3 Controlled Fake Shell

The fake shell provides an interactive SSH experience while avoiding real command execution. Common commands such as whoami, id, pwd, ls, uname -a, and ps return predefined responses.

For example, an attacker entering:

whoami

receives:

honeypot

The response is generated by the honeypot application and does not represent execution on the underlying host. This controlled, interactive fake-shell approach follows the same general design principle as established SSH/Telnet honeypots such as Cowrie, which have been used in comparative and containerized honeypot studies [3], [10].

This design is particularly important for safety because the attacker-controlled command string is stored and analyzed as data rather than being executed. The project implementation explicitly defines this as a safety-critical property.

4.4 Command Classification

The command logger records each submitted command together with its assigned category. A rule-based approach is used because it is deterministic and explainable. The current implementation records commands and categories separately from the detection engine.

4.5 Detection Engine

The detection engine evaluates suspicious command categories and produces alerts. Individual suspicious activities generate category-specific alerts, while combinations of different behavioral stages can result in a multi-stage alert.

For example, a sequence containing:

cat /etc/shadow

sudo -l

crontab -e

can represent progression through credential access, privilege escalation, and persistence behaviors.

4.6 Persistent Storage

SQLite is used to maintain structured security records. The principal tables are:

Table	Purpose
sessions	Stores SSH session information

authentication_attempts	Stores authentication activity
commands	Stores submitted commands and categories
alerts	Stores generated security alerts

Table.4.6.1 Persistent Storage

This structure provides a direct relationship between an SSH session, its authentication activity, submitted commands, and generated alerts.

4.7 Container Isolation

The honeypot runs inside Docker using security restrictions such as non-root execution, dropped capabilities, no-new-privileges, resource limitations, and a Docker-managed data volume, consistent with containerized honeypot isolation practices reported in prior work [1], [2], [10]. The architecture intentionally avoids host filesystem mounts and Docker socket exposure.

The isolation experiment attempted to create and inspect a marker file from the attacker-facing shell. The marker was absent both on the host and inside the container after the test, supporting the intended separation behavior.

The architecture separates interaction handling, command recording, detection, and data storage into distinct logical components. Incoming SSH connections are handled by the SSH server and authentication layer before an interactive fake shell is presented to the connected user. Commands received through the shell are recorded in the database and passed to the detection engine for behavioral classification and alert generation. The honeypot is deployed inside a Docker container to reduce the risk of interaction affecting the host environment.

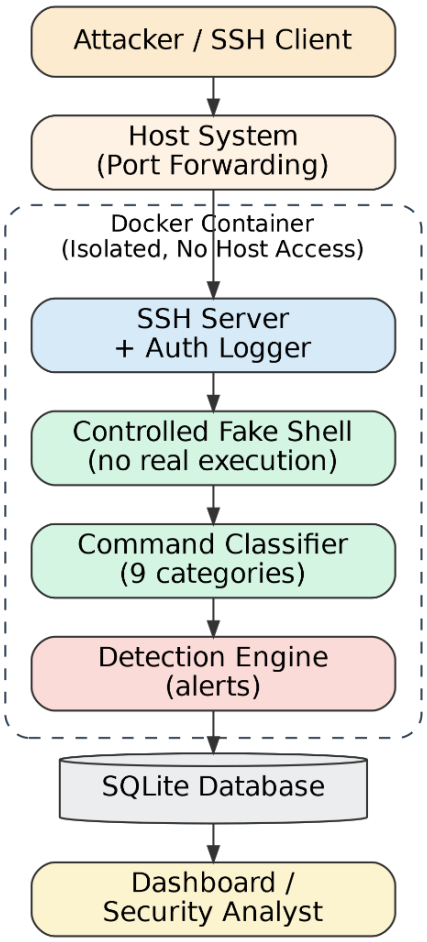

Fig. 4.7.1 Architecture Diagram

5. DISCUSSION

The implemented system demonstrates that a lightweight SSH honeypot can collect structured attacker activity while maintaining a controlled execution environment. The combination of a fake shell and Docker isolation allows commands to be observed without allowing the attacker to directly execute them on the host. One important observation is the difference between **data collection and security interpretation**. A honeypot may record a large number of commands, but raw command logs do not immediately explain the attacker's behavior. SSH-HoneyGuard addresses this by assigning commands to behavioral categories and generating alerts when suspicious activity is identified. The experimental data also demonstrates that attacker behavior is not limited to one category. The dataset contains reconnaissance, credential access, privilege escalation, persistence, network, download, destructive, and file-related activities. This makes multi-stage correlation useful because several individual commands may become more meaningful when considered together. The current implementation uses rule-based detection rather than machine learning, in contrast to recent AI-driven and reinforcement-learning-based honeypot analysis approaches [4], [6]. This provides an advantage for academic evaluation because each classification decision can be traced to a predefined rule. It also avoids requiring model training data or GPU infrastructure. The container isolation experiment further supports the design principle that the deception

environment should be separated from the host. The project architecture specifically avoids host filesystem mounting and Docker socket exposure, reducing the potential impact of attacker interaction. At the same time, the system should not be interpreted as a replacement for a production intrusion-detection platform. The current evaluation uses controlled experiments, a local deployment, and a predefined command rule set. These constraints provide a reproducible research environment but also define areas for future expansion.

6. RESULTS AND EVALUATION

The final experimental dataset stored by SSH-HoneyGuard contains **52 sessions, 50 authentication attempts, 409 commands, and 147 alerts**.

Table 1. Overall Research Dataset

Metric	Result
SSH sessions	52
Authentication attempts	50
Commands captured	409
Security alerts	147
HIGH alerts	126
MEDIUM alerts	21
Multi-stage alerts	11

Table.6.1 Overall Dataset

The command dataset was distributed across several behavioral categories.

Table 2. Command Classification Results

Category	Commands
Reconnaissance	163
Unclassified	81
Privilege escalation	38
Credential access	30
Persistence	28
Network	23
Download	21
Destructive	20
File activity	5
Total	409

Table.6.2 Command Classification Results

Of the 409 captured commands, **328 were classified into behavioral categories other than unclassified**, giving a classification coverage of approximately **80.20%**. The largest category was reconnaissance, with 163 commands. Privilege escalation, credential access, persistence, network, download, and destructive behaviors were also observed.

Table 3. Alert Distribution

Alert Type	Severity	Count
Suspicious command - privilege escalation	HIGH	38
Suspicious command - credential access	HIGH	29
Suspicious command - persistence	HIGH	28
Suspicious command - download	MEDIUM	21
Suspicious command - destructive	HIGH	20
Multi-stage activity	HIGH	11
Total		147

Table.6.3 Alert Distribution

The alert distribution shows that the majority of generated alerts were HIGH severity. This is mainly associated with commands classified as credential access, privilege escalation, persistence, and destructive activity. The system also generated **11 multi-stage activity alerts**. During the controlled multi-stage experiment, reconnaissance commands were followed by credential access, privilege escalation, and persistence activity. The system consequently generated separate category alerts as well as a multi-stage alert.

Table 4. Isolation Verification

Test	Result
Host marker before attack	Absent
Container marker before attack	Absent
Attacker attempted file creation	Yes
Host marker after attack	Absent
Container marker after attack	Absent
Successful host modification	No

Table.6.4 Isolation Verification

The restart-persistence experiment also demonstrated that the recorded database remained available after restarting the honeypot container. Before and after restart, the database contained 51 sessions, 49 authentication attempts, 378 commands, and 129 alerts. Overall, the experiments confirm the basic operational goals of the system: SSH interaction was captured, commands were categorized, suspicious activity generated alerts, multi-stage activity was identified, and the honeypot remained isolated from the host during the tested file-operation scenario.

7.IMPLEMENTATION AND OPERATIONAL EVIDENCE

This section presents direct evidence from the running SSH-HoneyGuard deployment, complementing the aggregate results reported in Section 6. The screenshots were captured from the actual containerized system described in Sections 3 and 4, including the terminal used to operate the honeypot and the read-only security dashboard built on top of the SQLite database.

Personally identifying terminal information (local username and machine name) has been redacted and replaced with a generic label; no other content was altered.

7.1 Security Dashboard

A read-only Streamlit dashboard was built on top of the SQLite database to provide a human-readable view of the same data summarized in Tables 1 through 3. The dashboard screenshots below were captured after further continuous operation of the honeypot beyond the dataset snapshot used for the experiments in Section 6; consequently, the totals visible in the dashboard (64 sessions, 462 commands, and 168 alerts at the time of capture) are somewhat larger than the values reported in Table 1, but the underlying category and severity distributions are consistent with Tables 2 and 3.

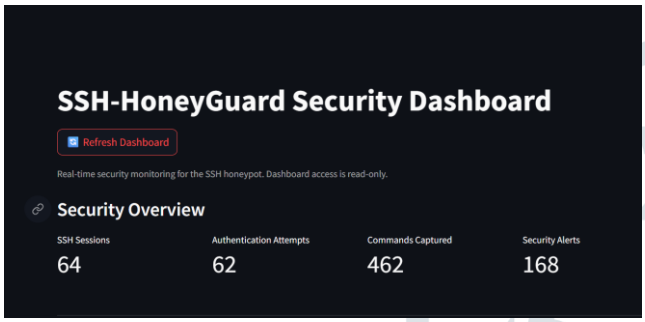

Fig. 7.1. Security dashboard overview showing headline session, authentication, command, and alert counts.

Fig. 7.2. Dashboard visualizations of commands by behavioral category and alerts by severity.

Fig. 7.2 mirrors the category distribution in Table 2 and the severity split in Table 3, with reconnaissance and unclassified commands forming the largest groups and HIGH-severity alerts substantially outnumbering MEDIUM-severity alerts.

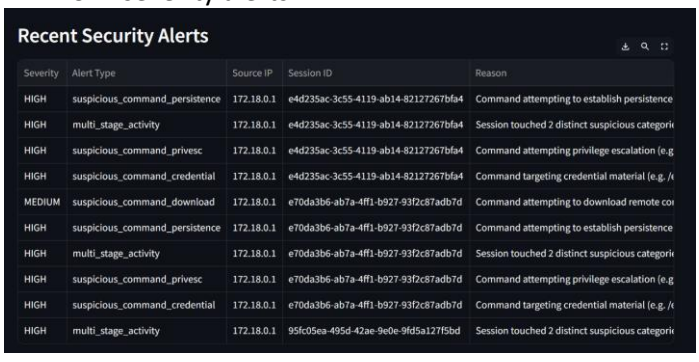

Fig. 7.3. Recent security alert's view, listing alert type, severity, source IP, session ID, and reason.

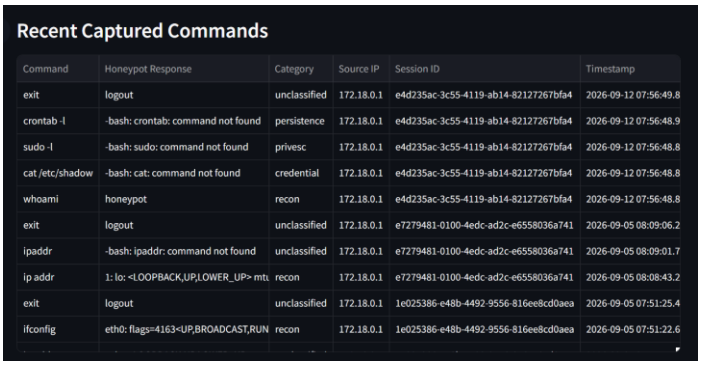

Fig. 7.4. Recent captured commands view, showing the command text, honeypot response, and assigned category.

Figs. 3 and 4 show the underlying per-event records behind the aggregate alert and command statistics: each alert is tied to a specific session and source address, and each command is shown alongside the exact (non-executed) response returned by the fake shell.

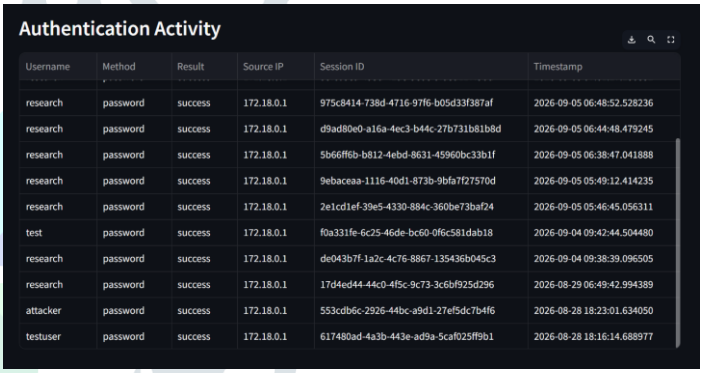

Fig. 7.5. Authentication activity view, showing per-attempt username, method, result, and timestamp.

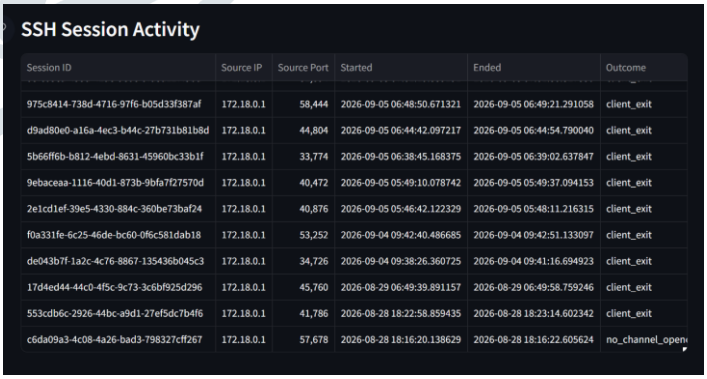

Fig. 7.6. SSH session activity view, showing per-session source port, start/end time, and outcome.

Figs. 5 and 6 correspond to the authentication_attempts and sessions tables described in Section 4.6, and provide the raw session-level detail from which the authentication and session counts in Table 1 are derived.

8. CONCLUSION

This paper presented SSH-HoneyGuard, a lightweight SSH honeypot designed to capture and analyze attacker

behavior within a controlled Docker environment. The system combines Paramiko-based SSH interaction, authentication logging, a controlled fake shell, command classification, behavioral detection, multi-stage activity detection, and SQLite-based storage. The experimental evaluation produced **52 sessions, 50 authentication attempts, 409 commands, and 147 alerts**. The captured commands covered several behavioral categories, including reconnaissance, credential access, privilege escalation, persistence, network activity, download activity, and destructive activity. The detection engine generated both individual suspicious-command alerts and multi-stage activity alerts. This supports the safety objective of treating attacker commands as input data rather than executing them on the host. The results indicate that SSH-HoneyGuard can provide more structured security information than simple SSH connection logging while retaining a relatively lightweight architecture. The rule-based detection mechanism also provides explainable decisions, which is useful for research analysis and academic evaluation.

9. FUTURE WORK

Several improvements can be considered for future versions of SSH-HoneyGuard.

First, the command-classification rules can be expanded to cover a larger range of Linux commands and more detailed attack techniques. This could reduce the proportion of unclassified commands observed during testing.

Second, the detection engine can be extended with behavioral scoring so that individual sessions receive an overall risk value based on the severity and sequence of observed activities.

Third, external threat-intelligence enrichment can be incorporated to associate source addresses with additional security information. This would allow locally captured behavior to be combined with external intelligence.

Fourth, a real-time dashboard can be used to present active sessions, command categories, alert severity, and multi-stage attack progression to analysts.

Finally, future experiments should use larger and more diverse datasets, including repeated attack scenarios and geographically distributed sources. This would provide stronger evidence for evaluating classification accuracy, detection performance, resource overhead, and generalization beyond the current controlled environment.

References

[1] T. Yu, Y. Xin, and C. Zhang, "HoneyFactory: Container-Based Comprehensive Cyber Deception Honeynet Architecture," *Electronics*, vol. 13, no. 2, 361, 2024.

[2] M. R. Amal and P. Venkadesh, "H-DOCTOR: Honeypot Based Firewall Tuning for Attack Prevention," *Measurement: Sensors*, vol. 25, 100664, 2023.

[3] C. Kelly, N. Pitropakis, A. Mylonas, S. McKeown, and W. J. Buchanan, "A Comparative Analysis of Honeypots on Different Cloud Platforms," *Sensors*, vol. 21, no. 7, 2433, 2021.

[4] P. Lanka, K. Gupta, and C. Varol, "Intelligent Threat Detection: AI-Driven Analysis of Honeypot Data to Counter Cyber Threats," *Electronics*, vol. 13, no. 13, 2465, 2024.

[5] L. Huang and Q. Zhu, "Adaptive Honeypot Engagement through Reinforcement Learning of Semi-Markov Decision Processes," arXiv:1906.12182, 2019.

[6] L. J. Möller, "An Adaptive Multi-Layered Honeynet Architecture for Threat Behavior Analysis via Deep Learning," arXiv:2512.07827, 2025.

[7] E. Feito-Casares, I. Gómez-Talal, and J.-L. Rojo-Álvarez, "Descriptor: Multi-Regional Cloud Honeynet Dataset (MURHCAD)," arXiv:2601.05813, 2026.

[8] M. Zambianco, C. Facchinetti, and D. Siracusa, "A Proactive Decoy Selection Scheme for Cyber Deception Using MITRE ATT&CK," arXiv:2404.12783, 2024.

[9] V. Valeros, M. Rigaki, and S. Garcia, "Attacker Profiling through Analysis of Attack Patterns in Geographically Distributed Honeypots," arXiv:2305.01346, 2023.

[10] V. S. Devi Priya and S. Sibi Chakkaravarthy, "Containerized Cloud-Based Honeypot Deception for Tracking Attackers," *Scientific Reports*, vol. 13, 1437, 2023.