A Multi-Phase Study on Honeypots: Testing, IDS, and AI Integration

Aline Hassan
Department of Electrical and Computer Engineering
American University of Beirut
Beirut, Lebanon
afh29@mail.aub.edu

Reem Arnaout
Department of Electrical and Computer Engineering
American University of Beirut
Beirut, Lebanon
ria42@mail.aub.edu

***Abstract*—The complexity of security threats and the diverse attacker models require innovative and continuously evolving security measures. Although not classified as intrusion detection systems (IDS), honeypots are important tools in security, designed to capture and log activities by trapping users. This paper presents an approach to testing several open source honeypots in a controlled environment. To be realistic in the testing process, IDS is integrated with the honeypots, enabling detection by IDS and trapping attackers by honeypots. Furthermore, the integration of artificial intelligence (AI) is added to investigate its importance in enhancing honeypot functionality.**

***Index Terms*—Honeypots, Intrusion Detection Systems, Artificial Intelligence, Services, Snort, GPT**

I. Introduction

A. Motivation

Nowadays, the nature of threats is changing due to many factors including the increase in attackers' expertise and the availability of easy-to-use intruder tools. So, there is a need to increase the security level to adapt with the evolvement of the threats. Honeypots have been praised for revealing new attack methods, patterns of attacker behavior, and various insights connected to defensive strategies [1]. However, honeypots alone are insufficient because they are not detection systems, and IDS alone struggles due to changing natures of attacks. The rapid growth of networking and computer attacks is increasingly challenging, leading to a high rate of false alerts[2]. Thus, integrating honeypots with IDS can enhance capabilities of both tools. To strengthen this approach, a multi-layered honeypot architecture can be employed. Such a design integrates various types of honeypots, such as low-interaction and high-interaction honeypots across multiple network layers, and deploys an IDS within the architecture. Each layer focuses on specific threats and offers tailored engagement to lure in attackers while isolating them from critical systems. Due to the importance of honeypots, their functionality should be enhanced. Since AI is being widely applied in several security domains, integrating it with honeypots has become crucial. AI-powered honeypots can be designed to adapt to evolving tactics of attackers and strategies that can evade the current security systems, making it challenging for the attacker to differentiate whether they are engaging with an actual system or a honeypot [3].

B. Objectives

Based on the motivation, this project aims to investigate the capabilities of different honeypots and explore their integration with IDS and AI. Thus the objectives are the following:

1) Deployment and testing of various types of honeypots that simulate vulnerable services to attract attackers
2) Analysis of data patterns and logs of tested honeypots
3) Integration of honeypots of IDS to enhance capabilities of both tools
4) Configuring and testing a multi-layered honeypot architecture
5) Development of AI-powered honeypot

II. Literature Review

A. Integration of IDS and Honeypots

Honeypots and IDS are both essential components in security. Integrating them into the same design can lead to a better security system with enhanced capabilities. For example, in [2], a system was designed to combine both IDS, and honeypot using a load balancer for their web server. This approach allowed them to detect attacks using IDS and collect more details about the intruder using honeypots. By doing so, updating IDS based on the data collected, could lead to better security. Similar work was done in [4], where the proposed system was to have a honeypot based-model for IDS in order to obtain the most useful information about the attacker and the attacks. Better work was done in [5], where they designed an automated rules-generation system for Snort using the logs of the honeypot after detecting malicious activity, leading to better and more specific rules. All this work and more focused on combining honeypot with IDS in order to enhance the performance of IDS by using the logs to generate new rules.

B. AI and Honeypots

Given the importance of honeypots in security, there is a constant need to upgrade the honeypots to adapt to new attacks. Integrating AI into honeypots can enhance their capabilities. For instance, in [3], researches showed that such integration can lead to better proactivity, adaptiveness, and insight. Further work was done in [6], where they used machine learning (ML) models as classifiers for detecting malicious activity. They proposed having honeypots that will save the activity of the users in log files, which will then be used to fine-tune the ML models leading to adaptive classification and, consequently, better detection. Recent advancements in AI powered honeypots have used Large Language Models (LLMs) due to their high potential in security. In [7] and [8], LLMs were integrated with honeypots to simulate the Linux Shell when attackers try to connect. They proved to be effective, especially in handling prolonged sessions with the attackers. More work is being done in this domain due to the advancements in AI domain that are greatly affecting other aspects of security.

C. *Multi-Layered Honeypots*

In the realm of cybersecurity research, a multi-layered approach has been widely recognized for its ability to enhance honeypot capabilities. In [9], researchers proposed a framework combining honeypots with network monitoring tools such as NTAs, NPM tools, and IDS. This layered architecture provided a robust defense mechanism by integrating early detection, real-time monitoring, and intrusion prevention. Further work in [10] explored honeypot based security framework deployed on AWS infrastructure. The study utilized the T-Pot honeypot setup due to its integration of multiple honeypots and monitoring tools. This in turn helped analyze attack patterns, detect targeted attacks, and log attacker behavior in real time. Building on the demand for diverse honeypot deployments, [11] introduced a multi-platform honeypot architecture which aimed at generating actionable cyber threat intelligence. The study implemented para-virtualization to deploy both low and high-interaction honeypots across IoT and industrial platforms.

III. Methodology and Design

1) Deployment and Testing of Honeypots
   - 5 Single Honeypots Deployment and Testing
     To deploy and test honeypots, 2 virtual machines (VMs) were used: one for the attacker and one for the honeypot deployment. The following 5 honeypots were chosen: Glutton, Cowrie, Heralding, Honeycomb and Galah. Each chosen honeypot was installed using the respective Github repository and configured according to the documentation provided for each honeypot. Attacks were conducted from the attacker VM, focused on the services simulated by the honeypots. Different types of attacks, such as port scans, SSH Brute-force attacks, and UDP flooding were used using Nmap. Each tested honeypot logs the activity differently from the others
   - Tpot Deployment and Testing
     In order to employ Tpot, 2 virtual machines (VMs) were used: one for configuring and running Tpot docker images and one for performing a simple scan to impersonate a malicious attacker. To run Tpot after primary installation, the network configurations had to be adjusted by creating a shared network for host independent honeypots and decreasing the stringency of iptables rules. Next, 26 docker images were pulled, built, and run using the docker-compose configuration file while ensuring that any conflicting service was temporarily stopped. Finally, the platform could be visualized through http://127.0.0.1:64297. Upon performing a single scan or attack, individual and cumulative honeypot responses alongside Suricata logs were produced in real-time.

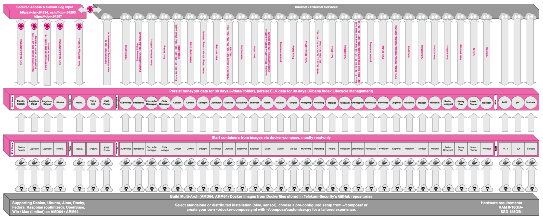
Secured Access & Sensor Log Input
https://<ip>:64294, ssh://<ip>:64295
https://<ip>:64297
Internet / External Services
Persist honeypot data for 30 days (~/data/ folder), persist ELK data for 30 days (Kibana Index Lifecycle Management)
ELK Stack
Elastic-Search
Logstash Input
Logstash Output
Kibana
Tools
NGINX
T-Pot Init
EWS Poster
Honeypots
ADBHoney
Beelzebub
CiscoASA Honeypot
Citrix Honeypot
Conpot
Cowrie
Ddospot
Dicompot
Dionaea
ElasticPot
Endlessh
Galah
Glutton
Go-pot
H0neytr4p
Heralding
Hellpot
Honeyaml
qHoneypots
Honeytrap
IPPHoney
Log4Pot
Mailoney
Medpot
Miniprint
Redis-Honeypot
Sentry Peer
Snare / Tanner
Wordpot
NSM
FATT
p0f
Suricata
Start containers from images via docker-compose, mostly read-only
Build Multi-Arch (AMD64, ARM64) Docker images from Dockerfiles stored in Telekom-Security's GitHub repositories
Supporting Debian, Ubuntu, Alma, Rocky, Fedora, Raspbian (optimized), OpenSuse, Win / Mac (limited) as AMD64 / ARM64.
Select standalone or distributed installation (hive, sensor), choose a pre-configured setup from ~/compose/ or create your own ~/docker-compose.yml with ~/compose/customizer.py for a tailored experience.
Hardware requirements
RAM 8-16GB+
SSD 128GB+

Fig. 1. Tpot's Technical Architecture. Source: tpotce

2) Integration of Honeypots with IDS
   To ensure realistic testing of honeypots, IDS was integrated with honeypot. The chosen IDS is Snort and the chosen honeypot is Glutton. Snort was configured to detect Session Initiation Protocol (SIP) flooding using specific rules. When such an attack is detected, Snort forwards the packets to Glutton to log the attacker activity, otherwise forward them to the legitimate server to connect normally. Figure 1 shows our design that allows us to mimic what typically happens in real world scenarios.

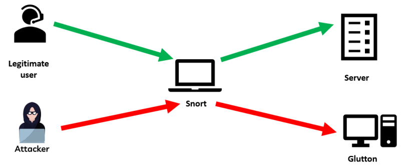

Fig. 2. Honeypot-IDS System

3) Development of GPT-Honeypot
   Due to the potential of LLMs in security domain, and the promising results they have shown, a simple honeypot was developed to simulate the performance of Telnet in Linux Shell. Since Telnet has various capabilities and prolonged sessions, integrating LLM to simulate the session was a logical solution to having static responses of Telnet honeypots. After several refinements of the prompts and formatting, the simple honeypot running on port 2228 was integrated with OpenAI API key to use GPT-4o-mini. Figure 2 shows the design on the GPT-Honeypot.

IV. Results and Analysis

A. *Results and Analysis*

1) Testing Honeypots

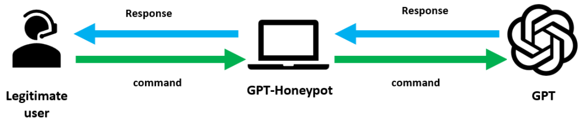

Fig. 3. GPT-Honeypot

Table I presents the results of testing different services of honeypots. Each honeypot generates different logs, which are In order to evaluate the tested honeypots, two metrics were used: Log Completeness and Deception Realism. For Log Completeness, below are the standard logs to find for each honeypot:

TABLE I
SUMMARY OF HONEYPOT SERVICES AND LOGGED INFORMATION

Honeypot	Services Offered	Services Tested	Information Logged
Glutton	SMTP, UDP, TCP, Telnet, SIP	SMTP, SIP, UDP, Telnet	Payload data, timestamps, sensor IDs
Galah	GPT-based honeypot, Web services	HTTP, HTTPS	Timestamp, request payload, HTTP responses, source-destination IPs/ports
Honeycomb	FTP, HTTP, LibSSH	FTP	Source IP/port, descriptions, credentials, Timestamp
Cowrie	SSH, Telnet	SSH	credentials, source IP, commands
Heralding	Telnet, SMTP, SSH, FTP, HTTP	Telnet, SMTP	Source and destination IPs and ports, session duration, session IDs, protocols used, authentication attempts
Tpot	16 Honeypots and 10 Cyber Tools	10 Honeypots and Suricata IDS	Source IP addresses, ports, protocols, session data, captured payloads and binaries, network metadata, and alerts generated from predefined rules.

- Glutton: Payload, timestamps, sensor IDs, Source IP/ports.
- Galah: Timestamp, request payload, HTTP responses, source and destination IP/ports
- Honeycomb: Source IP/port, descriptions, credentials, timestamp
- Cowrie: credentials, source IP, commands, timestamp
- Heralding: Source-destination IPs/ports, session duration, session IDs, protocols, attempts
- Tpot: Source IP addresses, ports, protocols, session data, captured payloads and binaries, network metadata, and alerts generated from predefined rules

Regarding Deception Realism, qualitative rating scale was used based on the interaction with the honeypots. Table II summarizes the evaluation. According to the results presented in the tables, honeypots have distinct performance in terms of logging information. Most of the honeypots provide comprehensive logging to be used later for enhancing security and detection systems

TABLE II
EVALUATION OF TESTED HONEYPOTS

Honeypot	Log Completeness	Deception Realism
Glutton	3/5	Good
Galah	1	Fair
Honeycomb	4/5	Good
Cowrie	1	Very Good
Heralding	1	Very Good
Tpot	11/17	Very Good

2) Honeypot-IDS Integration Results
The system forwards suspect SIP packets to the honeypot after being detected by Snort to log activity. According to Table I, the information logged for the SIP packets was reported by Glutton. So, for evaluation, same as Glutton for honeypot part and detected false positives by Snort. Automating the generation of Snort rules based on the honeypots' logs could lead to a better IDS.

3) Honeypot-AI Integration
After creating the simple honeypot and integrating GPT-4o-mini to simulate the Telnet functionalities, extensive testing was conducted. Since the API is stateless, a list of history commands and responses was passed as part of the prompt, leading to expected results. The honeypot was able to manage the full session. This lead to Log Completeness score of 1 and Excellent Deception Realism. Integrating LLMs with honeypots can lead to a better deception and trapping system.

V. CONCLUSION

Honeypots are crucial systems that provide insights into attackers' behaviors. Moreover, using all-in-one honeypots, often referred to as ”all-eating” honeypots, can lead to multi-layered logging. However, honeypots alone can't be used as independent systems because they don't differentiate between attackers and legitimate users. This is why having an IDS is essential for detection. However, also, an IDS alone can lead to high false positive rates if not configured correctly. Integrating honeypots with IDS can lead to an adaptive IDS by using the logs of suspicious packets to update the IDS configuration. Integrating LLMs into honeypots can also address the need for realistic responses by managing prolonged sessions, leading to better logging. As future work, combining all these elements will be done. Creating a multi-layered, LLM-powered honeypot and integrating it with Snort, while automating rules based on the honeypot's logs, would result in one of the best security systems.

REFERENCES

[1] P. Patel, A. Dalvi, and I. Sidddavatam, “Exploiting Honeypot for Cryptojacking: The other side of the story of honeypot deployment,” *2022 6th International Conference on Computing, Communication, Control and Automation (ICCUBEA*, pp. 1–5, Aug. 2022, doi: 10.1109/iccubea54992.2022.10010904.

[2] R. K. Singh and T. Ramajujam, “Intrusion detection system using advanced honeypots,” *arXiv (Cornell University)*, Jan. 2009, doi: 10.48550/arxiv.0906.5031.

[3] Shyamalendu Paul, Amitava Podder, Kaustav Roy, Anupama Sen, and Anindita Chakraborty, “Exploring the Impact of AI-based Honeypots on Network Security”, *kuey*, vol. 30, no. 6, pp. 251–258, Jun. 2024.

[4] J. R. Kondra, S. K. Bharti, S. K. Mishra, and K. S. Babu, “Honeypot-based intrusion detection system: A performance analysis,” *International Conference on Computing for Sustainable Global Development*, pp. 2347–2351, Mar. 2016.

[5] A. Sagala, ”Automatic SNORT IDS rule generation based on honeypot log,” *2015 7th International Conference on Information Technology and Electrical Engineering (ICITEE)*, Chiang Mai, Thailand, 2015, pp. 576-580, doi: 10.1109/ICITEED.2015.7409013.

[6] R. Vishwakarma and A. K. Jain, ”A Honeypot with Machine Learning based Detection Framework for defending IoT based Botnet DDoS Attacks,” *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, Tirunelveli, India, 2019, pp. 1019-1024, doi: 10.1109/ICOEI.2019.8862720.

[7] C. Guan, G. Cao and S. Zhu, ”HoneyLLM: Enabling Shell Honeypots with Large Language Models,” *2024 IEEE Conference on Communications and Network Security (CNS)*, Taipei, Taiwan, 2024, pp. 1-9, doi: 10.1109/CNS62487.2024.10735663.

[8] M. Sladić, V. Valeros, C. Catania and S. Garcia, ”LLM in the Shell: Generative Honeypots,” *2024 IEEE European Symposium on Security and Privacy Workshops (EuroSPW)*, Vienna, Austria, 2024, pp. 430-435, doi: 10.1109/EuroSPW61312.2024.00054.

[9] T. Shivaprasad, A. S. Moulya, and N. Guruprasad, “Enhancing Network Security through a Multi-layered Honeypot Architecture with Integrated Network Monitoring Tools,” *2022 9th International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 473–477, Feb. 2024, doi: 10.23919/indiacom61295.2024.10498895.

[10] P. Subhash, M. Qayyum, C. L. Varsha, K. Mehernadh, J. Sruthi, and A. Nithin, “A security framework for the detection of targeted attacks using Honeypot,” in *Lecture notes in networks and systems*, 2024, pp. 183–192. doi: 10.1007/978-981-99-9704-6.

[11] S. Kumar, B. Janet and R. Eswari, ”Multi Platform Honeypot for Generation of Cyber Threat Intelligence,” *2019 IEEE 9th International Conference on Advanced Computing (IACC)*, Tiruchirappalli, India, 2019, pp. 25-29, doi: 10.1109/IACC48062.2019.8971584.

LINKS

A. Video Link

B. Github Repository

CONTRIBUTION

Section	**Member**
Abstract	Aline
Motivation	Aline and Reem
Objectives	Reem
Literature Review (Integration of IDS and Honeypots and AI and Honeypots)	Aline
Literature Review (Multi-Layered Honeypots)	Reem
Methodology and Testing (Deployment and Testing of Honeypots)	Reem
Methodology and Testing (Integration of Honeypots with IDS and Development of GPT Honeypot)	Aline
Results and Analysis	Aline and Reem
Conclusion and References	Aline and Reem