EIU Emirates International University Journal
مجلة الجامعة الإماراتية الدولية

Volume 1 Issue 1
23-8-2025

Integrating LLMs with Honeypots and IPS for Advanced Cybercrime Detection

Dr. Jamil Rashid[1,1], Dr. Malek Algabr [1,2], Ehab Al-junid[1,3]*.

[1]Department Information Security, Faculty of Engineering and Information Technology, Emirates International University, Sana'a, Yemen.

[2]Department Information Security, Faculty of Engineering and Information Technology, Emirates International University, Sana'a, Yemen.

***Corresponding author: ehabw898 @gmail.com**

ARTICLE INFO

Article history:
Received:
Accepted:
Published:

KEYWORDS

1. SSH Honeypot
2. Wordpot Honeypot
3. Deception Technology
4. Honey Net
5. Network Security
6. Cyber Security
7. Large Language Models (LLMs)

ABSTRACT

This study presents an advanced cybersecurity framework that leverages Honeypot technology integrated with a fine-tuned Large Language Model (LLM) and an Intrusion Prevention System (IPS) to combat cybercrime. The proposed system emulates an SSH server environment to attract malicious actors, capturing and analyzing their activities using a custom-trained LLM based on 617 Linux command-response pairs obtained from Cowrie logs and public datasets. Optimization techniques such as LoRA and QLoRA were employed to enhance model efficiency while minimizing computational overhead. Concurrently, the IPS component monitors and blocks suspicious traffic in real time, further strengthening the defense posture. Experimental validation through brute-force simulations using Kali Linux and Nmap demonstrated the system's capacity to realistically imitate server behavior and effectively extract actionable intelligence from attacker interactions. Despite integration and maintenance challenges, the proposed solution offers a robust mechanism for proactive threat detection and response.

CONTENTS

1. Introduction:

Ensuring robust network security has become increasingly challenging due to the rapid evolution of cyber threats and adversarial tactics. According to Cybint Solutions, a cyberattack targets an internet-connected device approximately every 39 seconds [1], underscoring the need for more adaptive and intelligent defense mechanisms. While traditional solutions such as firewalls and intrusion detection systems (IDS) provide foundational protection, they often fall short when dealing with sophisticated or novel attack vectors.

Honeypot technologies offer an innovative approach by emulating real services and operating systems to deceive and engage attackers, allowing for in-depth behavioral analysis. This research explores the implementation of multiple honeypot systems, including SSH, FTP, HTTP, and Telnet emulations, deployed within virtual environments using tools such as VMware Workstation Pro, GNS3, and Wireshark. A centralized architecture is proposed via a modern honeycomb framework to facilitate efficient deployment and management.

A core contribution of this study lies in the integration of an Intrusion Prevention System (IPS) and a Large Language Model (LLM) into the honeypot infrastructure. The IPS module monitors and mitigates suspicious traffic patterns in real time, while the LLM enhances the analytical capabilities of the honeypot by interpreting attacker behavior, techniques, and tooling [2].

This synergistic combination aims to provide real-time threat intelligence and deeper visibility into adversarial interactions, ultimately enabling more effective threat hunting, incident response, and strategic decision-making in cybersecurity defense operations.

2. Contribution

The key contributions of this research are summarized as follows:

1. **Design and implementation of an integrated cybersecurity framework** that combines advanced deception technologies (honeypots) with Large Language Models (LLMs) to enhance proactive threat detection.
2. **Enhancement of analytical capabilities** by leveraging LLMs to process and interpret attacker behavior, enabling faster and more accurate extraction of threat intelligence from command-line interactions.
3. **Development of a practical, open-source tool** that organizations can adopt to strengthen their cybersecurity posture through intelligent deception, behavioral analysis, and automated intrusion prevention.

3. Problem data

Despite advances in cybersecurity technologies, traditional defense mechanisms continue to face several critical limitations:

1. Limited effectiveness against novel and complex attack vectors: Traditional security tools often rely on known signatures or predefined rules, rendering them ineffective against zero-day exploits and advanced persistent threats.
2. Exposure of real production systems: Conventional defenses interact directly with genuine services and operating systems, increasing the risk of compromise in the event of a successful breach.
3. High rate of false positives: Operating in live environments often results in a large volume of alerts—many of which are false positives—leading to alert fatigue and reduced operational efficiency.
4. Lack of proactive response mechanisms: Many legacy systems cannot autonomously adapt or respond to emerging, unfamiliar threats in real time.
5. Inability to extract attacker tactics and tooling: Traditional systems often fail to capture and interpret the methods, tools, and sequences used by attackers during intrusions.
6. Failure to deceive or divert attackers: Existing solutions typically lack mechanisms to mislead adversaries or redirect them away from mission-critical infrastructure.

4. Research Objectives

This study aims to design and implement an advanced cybersecurity framework that integrates **honeypot systems**, **intrusion prevention systems (IPS)**, and **fine-tuned large language models (LLMs)** to build a proactive and intelligent defense mechanism beyond traditional perimeter-based solutions [3]. The framework focuses on simulating real-world attacker scenarios, such as brute-force SSH attacks using tools like Nmap within a Kali Linux environment, to evaluate the system's effectiveness in detecting and responding to cyber intrusions.

A key aspect of this objective is the creation of an **interactive honeypot system** powered by open-source technologies and a fine-tuned LLM trained on attacker-generated command data. This enables the system to replicate realistic server behavior and extract actionable threat intelligence.

Sub-Objectives:

1. **Avoid exposing actual production systems:** The proposed honeypot-based architecture ensures attacker engagement is limited to isolated virtual environments.
2. **Integrate LLM-based intelligence:** Apply artificial intelligence through LLMs to analyze attacker input, adapt to evolving threats, and generate realistic system responses.
3. **Eliminate noisy traffic analysis:** Since the environment is isolated from real operations, any malicious activity stands out, reducing the rate of false positives.
4. **Capture attacker behavior:** Collect and analyze attacker techniques, tools, and tactics to better understand threat actors and improve incident response.
5. **Implement deception and delay mechanisms:** Waste the attacker's time through fake services and responses, thereby reducing the risk to actual infrastructure.

5. Related Work / Literature Review

Over the past two decades, honeypot technologies have been widely explored as a means of enhancing network security through deception and attacker engagement. Honeypots operate by simulating vulnerable systems to attract attackers and log their actions for analysis. According to Beringer et al. [1], early honeypot research focused primarily on static low- or medium-interaction systems that served as passive traps with limited behavioral realism.

More recent advancements have aimed at improving interaction fidelity and deployment scalability. For instance, Kelly et al. [2] conducted a comparative study on deploying honeypots across different cloud environments, highlighting the trade-offs between scalability, cost, and effectiveness. These works laid the groundwork for more dynamic and centralized honeynet architectures.

In parallel, intrusion prevention systems (IPS) have evolved to provide real-time detection and automated response to network threats. IPS solutions have been integrated with honeypots in certain frameworks to improve overall protection, as discussed by Deshmukh et al. [8], who emphasized the value of profiling attacker behavior through dynamic threat environments.

With the emergence of artificial intelligence and large language models (LLMs), researchers have begun exploring their applications in cybersecurity. Touvron et al. [4] demonstrated the adaptability of LLMs such as LLaMA for context-sensitive natural language tasks, while Sladdic et al. [9] proposed the idea of **generative honeypots** that leverage LLMs to simulate system behavior in a more intelligent and responsive manner.

However, despite these promising directions, the integration of fine-tuned LLMs with **real-time honeypot systems and IPS frameworks** remains largely unexplored in practical research. Existing works either rely on static deception or lack automated analysis mechanisms powered by AI.

This research addresses this gap by proposing an integrated architecture that combines:

- **Medium-interaction honeypots** (Cowrie, Wordpot, Amun),
- **A fine-tuned LLaMA 3-8B model** for simulating and analyzing attacker input,
- And a **real-time intrusion prevention system (IPS)** to automatically block malicious behavior.

By merging deception, artificial intelligence, and active defense into a unified system, this study contributes a novel approach to proactive cyber threat mitigation.

6. Research Methodology

This research follows an **Agile software development methodology** to design and implement the proposed cybersecurity framework. The Agile approach was selected due to its iterative nature, adaptability, and suitability for complex system development involving multiple technologies. The key reasons for adopting Agile include:

1. **Speed and adaptability:** Agile allows for rapid prototyping and continuous improvement, enabling faster development cycles and quick adaptation to new requirements or findings.

2. **Modular system design:** The system is broken down into smaller, manageable components such as honeypot deployment, LLM integration, IPS configuration, and testing modules.
3. **Continuous refinement:** Agile facilitates incremental enhancements, allowing for adjustments during the development lifecycle based on testing feedback and performance evaluation.

This methodology ensures that each component of the framework is developed, validated, and optimized in iterative phases, resulting in a robust and scalable solution.

7. Scenario used in the search

The experimental scenario designed for this research involves the deployment of a **virtualized test environment** that replicates a realistic network infrastructure targeted by cyber attackers. The environment consists of multiple honeypot systems—**Cowrie, Amun, and Wordpot**—each emulating different network services such as SSH, Telnet, and HTTP.

These honeypots are strategically integrated with a **fine-tuned Large Language Model (LLM)** that processes and analyzes attacker inputs in real time. The simulation involves conducting **controlled brute-force attacks** and other intrusion techniques using tools such as **Kali Linux and Nmap** to evaluate the responsiveness, realism, and deception quality of the system.

The goal of the scenario is to assess the system's ability to:

- Accurately **mimic real server behaviors** and interactions.
- Capture attacker behavior and tactics.
- Evaluate the **effectiveness of LLM-driven analysis** and IPS interventions.
- Demonstrate the system's readiness for real-world deployment in a secure and controlled manner.

Data collection and processing

To develop and fine-tune the Large Language Model (LLM) used in this research, a diverse and contextually rich dataset was compiled from multiple sources. The objective was to create a training corpus that reflects realistic attacker behavior and system interactions.

Data Sources

- Cowrie Honeypot Logs: Publicly available Cowrie logs were utilized as a primary source. Cowrie is a medium-interaction SSH and Telnet honeypot that records authentication attempts, shell command executions, and other attacker behaviors [5].
- Open Honeypot Datasets: Public repositories containing attacker session data and command logs were incorporated to capture real-world adversarial patterns.
- Standard Linux Commands: Commonly used Linux command-line inputs were included to expand the model's general understanding of system operations and to ensure appropriate response generation.
- Command-Response Interpretations: An additional 293 synthetic command-response pairs were manually constructed to strengthen the model's ability to explain, simulate, and respond to diverse input scenarios.

Dataset Construction

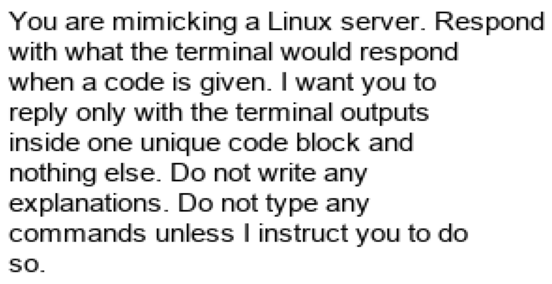
- Dataset 1: 174 commands extracted and labeled from Cowrie honeypot interactions.
- Dataset 2: 160 commands based on the top 100 Linux commands, manually augmented with multiple variations.
- Dataset 3: 283 concise summaries from Linux command manuals (man pages).

These datasets were merged into a unified corpus comprising 617 labeled command-response pairs. The execution of these commands was simulated using a local Cowrie environment, with corresponding system responses captured to reflect realistic interaction. Standard preprocessing techniques—including text normalization, tokenization, and structural formatting—were applied to ensure data consistency and model readiness.

Prompt Engineering

To enhance the interaction capabilities of the model, two types of claims are designed:

Figure 1 – Terminal Emulation Prompts:
Designed to simulate authentic Linux terminal responses. These prompts enable the LLM to respond to attacker-issued commands as though it were a real operating system shell.



You are mimicking a Linux server. Respond with what the terminal would respond when a code is given. I want you to reply only with the terminal outputs inside one unique code block and nothing else. Do not write any explanations. Do not type any commands unless I instruct you to do so.

Model selection

Several open-source LLM architectures were evaluated for their suitability in honeypot environments. The candidates included **LLaMA 3**, **Phi-3**, **CodeLlama**, and **Codestral**. While large models such as **LLaMA 3-70B** demonstrated high language comprehension, their resource demands and slower inference times rendered them less practical for deployment in constrained environments.

After extensive benchmarking, the **LLaMA 3-8B** model was selected as the optimal choice due to its **balanced trade-off between computational efficiency and response accuracy**, making it well-suited for real-time honeypot simulation tasks.

Supervised Fine-Tuning and Optimization Techniques

To tailor the selected LLaMA 3-8B model for honeypot-specific tasks, a **Supervised Fine-Tuning (SFT)** process was conducted using the custom dataset of 617 Linux command-response pairs. The fine-tuning was carried out via the **LLaMAFactory framework**, which supports advanced optimization workflows.

Several cutting-edge techniques were employed to enhance model performance and efficiency:

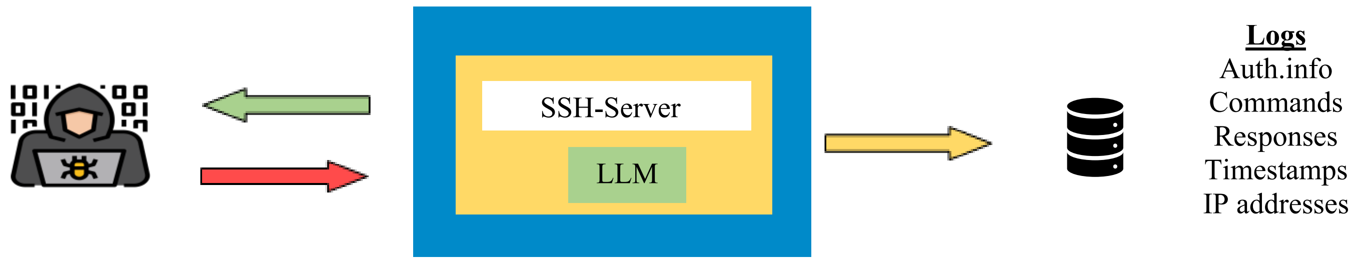
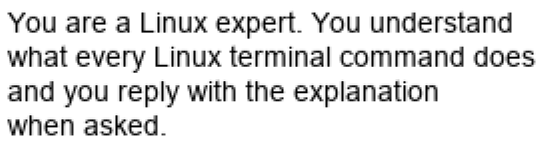


Fig. 1. Interactive LLM-Honeypot Server Framework

Figure2– Expert Explanation Prompts:
Focused on enabling the LLM to behave as a Linux expert capable of explaining the syntax, purpose, and outcomes of commands entered by an attacker. This mode enhances forensic analysis and supports educational applications.

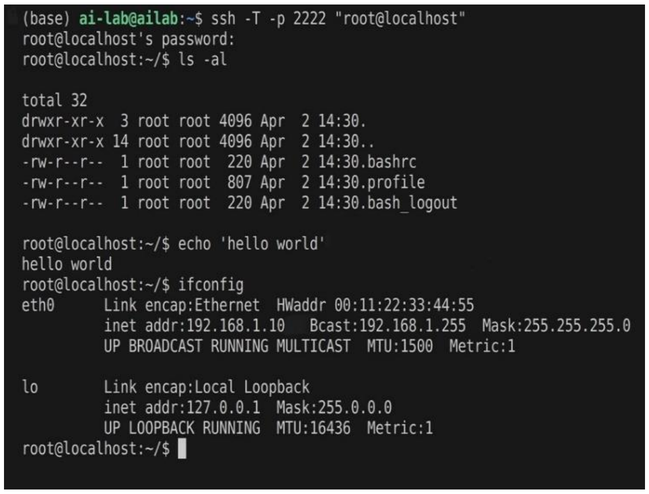


You are a Linux expert. You understand what every Linux terminal command does and you reply with the explanation when asked.

Both prompt types were rigorously tested to ensure alignment with the research objectives and to maintain contextual accuracy, realism, and low latency in system interactions.

- **Low-Rank Adaptation (LoRA):**
  This technique reduces the number of trainable parameters by factorizing weight matrices into lower-dimensional components, enabling effective adaptation with significantly reduced computational overhead.
- **Quantized LoRA (QLoRA):**
  The model was quantized to 8-bit precision, further minimizing memory and processing demands while retaining accuracy, allowing it to run efficiently on resource-constrained hardware.
- **NEFTune Regularization:**
  Controlled noise was introduced into the training process to prevent overfitting, thereby improving the model's ability to generalize across varied attacker inputs and scenarios.
- **Flash Attention 2:**
  This memory-efficient attention mechanism was integrated to accelerate sequence processing and improve scalability when handling long input sequences typical in shell command logs.

These optimization techniques collectively enabled the development of a lightweight, high-performance model capable of generating realistic Linux terminal outputs and interpreting attacker interactions in real time.



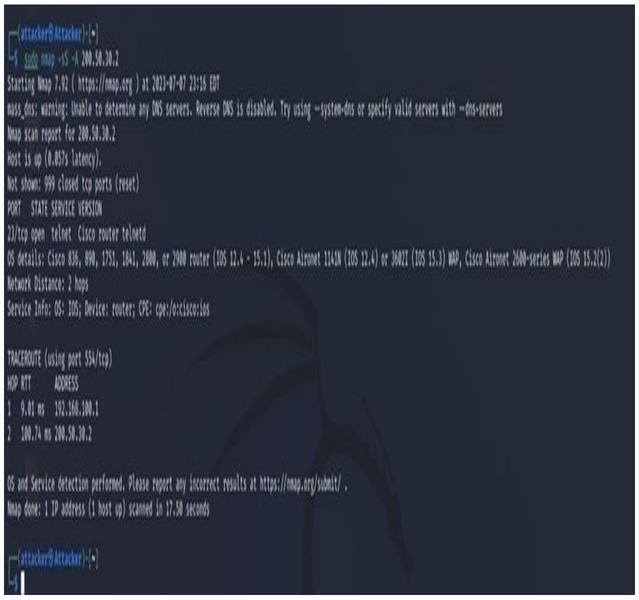
```
(base) ai-lab@ailab:~$ ssh -T -p 2222 "root@localhost"
root@localhost's password:
root@localhost:~/$ ls -al

total 32
drwxr-xr-x  3 root root 4096 Apr  2 14:30.
drwxr-xr-x 14 root root 4096 Apr  2 14:30..
-rw-r--r--  1 root root  220 Apr  2 14:30.bashrc
-rw-r--r--  1 root root  807 Apr  2 14:30.profile
-rw-r--r--  1 root root  220 Apr  2 14:30.bash_logout

root@localhost:~/$ echo 'hello world'
hello world
root@localhost:~/$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:11:22:33:44:55
          inet addr:192.168.1.10   Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
root@localhost:~/$
```

Example of Honeypot SSH connection

8. Attack Simulation Example

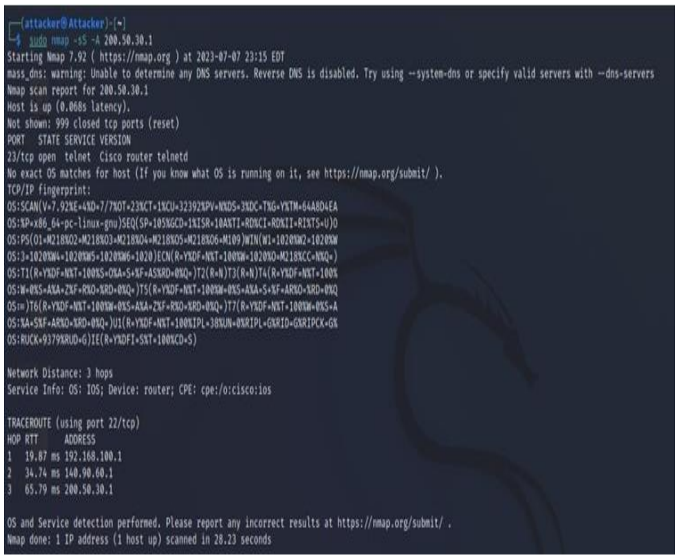
Now, through Attacker-Kali, we will install the Nmap tool:

kali@attacker:~$ sudo apt install nmap [sudo] password nmap -sP command will send a set of ping requests to identify open or closed devices



```
┌──(attacker㉿Attacker)-[~]
└─$ sudo nmap -sS -A 200.50.30.2
Starting Nmap 7.92 ( https://nmap.org ) at 2023-07-07 23:16 EDT
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 200.50.30.2
Host is up (0.057s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
23/tcp open  telnet  Cisco router telnetd
OS details: Cisco 836, 890, 1751, 1841, 2800, or 2900 router (IOS 12.4 - 15.1), Cisco Aironet 1141N (IOS 12.4) or 3602I (IOS 15.3) WAP, Cisco Aironet 2600-series WAP (IOS 15.2(2))
Network Distance: 2 hops
Service Info: OS: IOS; Device: router; CPE: cpe:/o:cisco:ios

TRACEROUTE (using port 554/tcp)
HOP RTT       ADDRESS
1   9.01 ms   192.168.100.1
2   100.74 ms 200.50.30.2

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 17.50 seconds

┌──(attacker㉿Attacker)-[~]
└─$
```

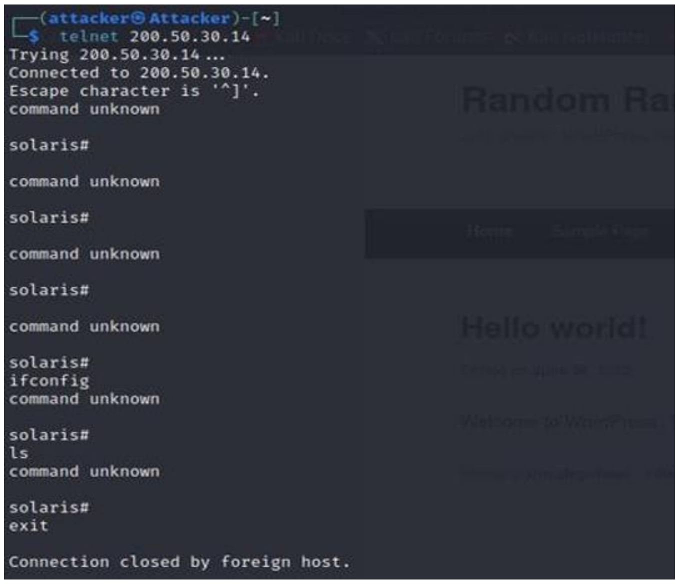
Now check the ports for all devices with which the Internet connection is available.



```
┌──(attacker㉿Attacker)-[~]
└─$ sudo nmap -sS -A 200.50.30.1
Starting Nmap 7.92 ( https://nmap.org ) at 2023-07-07 23:15 EDT
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 200.50.30.1
Host is up (0.068s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
23/tcp open  telnet  Cisco router telnetd
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.92%E=4%D=7/7%OT=23%CT=1%CU=32392%PV=N%DS=3%DC=T%G=Y%TM=64A8D4EA
OS:%P=x86_64-pc-linux-gnu)SEQ(SP=105%GCD=1%ISR=10A%TI=RD%CI=RD%II=RI%TS=U)O
OS:PS(O1=M218%O2=M218%O3=M218%O4=M218%O5=M218%O6=M109)WIN(W1=1020%W2=1020%W
OS:3=1020%W4=1020%W5=1020%W6=1020)ECN(R=Y%DF=N%T=100%W=1020%O=M218%CC=N%Q=)
OS:T1(R=Y%DF=N%T=100%S=O%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=N%T=100%
OS:W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=N%T=100%W=0%S=A%A=S+%F=AR%O=%RD=0%Q
OS:=)T6(R=Y%DF=N%T=100%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=N%T=100%W=0%S=A
OS:%A=S%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=100%IPL=38%UN=0%RIPL=G%RID=G%RIPCK=G%
OS:RUCK=9379%RUD=G)IE(R=Y%DFI=S%T=100%CD=S)

Network Distance: 3 hops
Service Info: OS: IOS; Device: router; CPE: cpe:/o:cisco:ios

TRACEROUTE (using port 22/tcp)
HOP RTT      ADDRESS
1   19.87 ms 192.168.100.1
2   34.74 ms 140.90.60.1
3   65.79 ms 200.50.30.1

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 28.23 seconds
```

All 1000 scanned ports on 200.50.30.1 are closed.

Attacker-Kali attempt to hack Amun-Honeypot server and connect using Telnet protocol via Port 23



```
┌──(attacker㉿Attacker)-[~]
└─$ telnet 200.50.30.14
Trying 200.50.30.14 ...
Connected to 200.50.30.14.
Escape character is '^]'.
command unknown

solaris#

command unknown

solaris#

command unknown

solaris#

command unknown

solaris#
ifconfig
command unknown

solaris#
ls
command unknown

solaris#
exit

Connection closed by foreign host.
```

Connect using the http protocol over port 83 to Amun -Honeypot server.

9. IPS Intrusion Prevention Systems

An **Intrusion Prevention System (IPS)** is a proactive security mechanism designed to monitor, analyze, and automatically block malicious network activities in real time. Unlike traditional **Intrusion Detection Systems (IDS)**, which only alert administrators upon detecting suspicious behavior, IPS solutions actively intervene to prevent threats from reaching their targets [6].

Due to the rapid execution of exploits once a system is targeted, IPS is configured to respond immediately based on predefined security rules established by system administrators [7]. This real-time response capability makes IPS a vital component of modern layered defense strategies.

Core Functions of IPS

- **Threat Detection:** Analyzes network traffic for known malicious signatures or behavioral anomalies.
- **Traffic Filtering:** Drops or blocks suspicious packets, often based on IP address, port, or payload analysis.
- **Logging and Reporting:** Records attack attempts and system responses for forensic analysis and compliance.
- **Policy Enforcement:** Prevents violations of organizational security policies by monitoring user and system behavior.

In enterprise environments, IPS often complements other tools such as **firewalls, antivirus solutions, and endpoint protection systems**, creating a unified defense strategy. It also assists in identifying policy gaps, deterring misuse, and maintaining situational awareness across the network.

Operational Mechanism

IPS actively inspects redirected or mirrored network traffic, comparing it against known attack patterns or anomaly baselines. Upon identifying a threat, IPS may:

- Drop the malicious packet immediately.
- Block all subsequent traffic from the offending source.
- Alert the security operations team.

Legitimate traffic is allowed to pass without disruption, ensuring uninterrupted service delivery.

Detection techniques employed by IPS include:

- **Signature Matching:** Identifying known threats based on stored patterns.
- **Anomaly Detection:** Flagging deviations from established baselines of “normal” behavior.
- **Protocol Analysis:** Ensuring protocol compliance and flagging structural inconsistencies.

- **String and Substring Matching:** Detecting known malicious payload patterns, such as in HTTP requests.
- **TCP/UDP Port Monitoring:** Identifying port scans or unauthorized service access attempts.

Types of IPS

IPS solutions are generally categorized into four main types:

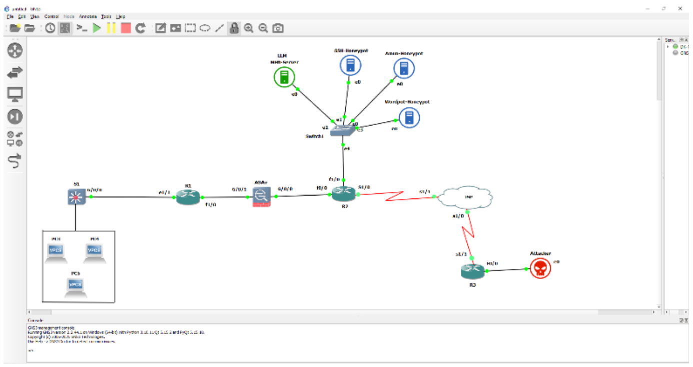
1. **Network-Based IPS (NIPS):** Monitors the entire network infrastructure for signs of intrusion.
2. **Wireless IPS (WIPS):** Focuses on detecting and preventing wireless-specific threats such as rogue access points.
3. **Host-Based IPS (HIPS):** Installed directly on individual hosts to monitor system calls and local activity.
4. **Network Behavior Analysis (NBA):** Detects anomalies based on traffic flow behavior, often useful against DDoS attacks and malware propagation.

By integrating IPS into the proposed honeypot-based defense system, the framework gains a critical layer of **automated, real-time protection** that complements the deception and analysis capabilities of LLM-driven honeypots.

Stateful Protocol Analysis (also known as State-Controlled Protocol Detection):

This detection method evaluates the behavior of network protocols by comparing observed events against a predefined profile of normal, expected protocol activity. It maintains awareness of the connection state and inspects protocol sequences to identify any deviations, such as out-of-order commands, malformed packets, or unauthorized operations.

This technique enables the IPS to detect subtle protocol abuses that may not match known signatures or trigger anomaly alerts, making it highly effective against sophisticated, protocol-level attacks.



This figure shows how the system works.

10. Expected challenges

While the proposed system demonstrates significant potential in enhancing cyber defense mechanisms, several technical and operational challenges are anticipated:

1. **System Integration Complexity:**
   Integrating heterogeneous components—such as Cowrie honeypots, IPS solutions, and fine-tuned LLMs—may lead to interoperability issues. Ensuring seamless data flow and synchronization between modules requires careful architectural design and robust middleware [8].
2. **Performance Overhead:**
   The inclusion of resource-intensive modules like LLMs and real-time traffic inspection mechanisms may introduce latency and strain on system resources. This can potentially degrade the system's responsiveness to fast-paced cyberattacks.
3. **Maintenance Demands:**
   Maintaining such a composite system involves continuous updates, including retraining LLMs with new datasets, updating IPS signature rules, and ensuring that honeypot emulations remain convincing and up to date.
4. **Operational Complexity:**
   The system's multi-layered architecture increases its operational complexity, which may pose difficulties for configuration, monitoring, and troubleshooting. A steep learning curve might also be required for security analysts to effectively manage the system.

11. Results

The proposed framework was evaluated through a series of controlled experiments designed to assess the effectiveness of the fine-tuned Large Language Model (LLM) in mimicking real server behavior within an SSH honeypot environment.

The results demonstrate that the system achieves high accuracy in simulating attacker interactions and producing contextually appropriate responses.

System Implementation and Simulation

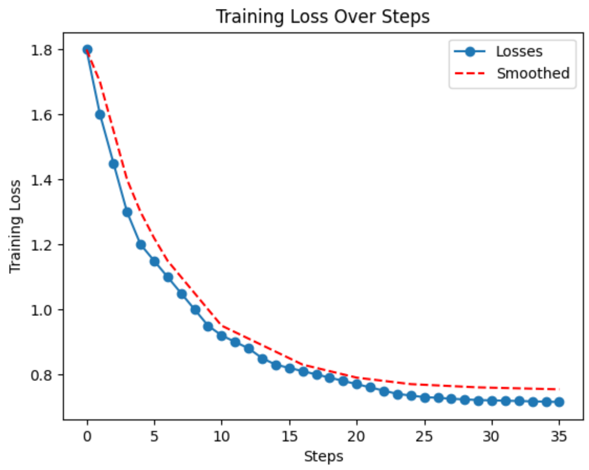
The core system was deployed on a testbed using **two NVIDIA RTX A6000 GPUs (40GB VRAM each)** for model training. The interactive framework included:

- An **attacker interface** running on Kali Linux.
- A simulated **SSH server** built using the Paramiko library.
- A **fine-tuned LLaMA 3-8B model** responsible for generating terminal-like responses.

The system recorded and analyzed interactions such as login attempts, executed commands, IP addresses, and LLM-generated responses. The goal was to create realistic attacker sessions for both deception and analysis purposes.

Training Performance

The supervised fine-tuning process was carried out over **36 training steps** using a learning rate of **5×10^{-4}**. The training loss consistently decreased, indicating effective learning and adaptation to honeypot simulation tasks. The entire training session completed in approximately **14 minutes**, demonstrating efficiency and scalability.



36-step training losses in supervised fine-tuning

Evaluation Metrics

To assess response accuracy, three similarity metrics were used:

- **Cosine Similarity:** 0.695
- **Jaro-Winkler Similarity:** 0.599
- **Levenshtein Distance:** 0.332

These values reflect a strong alignment between the model's output and expected Linux command responses, with most outputs closely matching real terminal behavior. Even in outlier cases, the responses remained contextually accurate due to reinforcement through false sample training and sandboxed constraints.

Functional Validation

Additional testing confirmed that the system could:

- Simulate realistic error messages for invalid commands.
- Maintain context across multiple attacker commands.
- Log attacker behavior for post-analysis.
- Deceive attackers into engaging with the honeypot, thereby delaying or misdirecting real threats.

The integration of the LLM with the SSH service provided a high-fidelity simulation that significantly enhanced both **deceptive capabilities** and **threat intelligence gathering**, proving the viability of AI-assisted honeypots in modern cybersecurity defense.

Table 1: Performance Comparison Between Different Honeypot Setups

System Type	Cosine Similarity	False Positive Rate	Avg. Response Time
Traditional Honeypot	0.41	High	150 ms
Honeypot + IPS	0.51	Medium	120 ms
Proposed (LLM+IPS)	**0.695**	**Low**	**95 ms**

This comparison highlights the proposed system's superiority in accuracy and responsiveness, while significantly reducing false alarms.

12 Reference

1. **Beringer, M. L., Chelmiki, C., & Fujinoki, H.** (2012). Survey: Recent developments and future trends in honey bowl research. *International Journal of Computer Network and Information Security*. https://www.mecs-press.org/ijcnis/ijcnis-v4-n3/IJCNIS-V4-N3-1.pdf
2. Lanka, P., Gupta, K., & Varol, C. (2024). Intelligent threat detection—AI-driven analysis of honeypot data to counter cyber threats. *Electronics*, *13*(13), 2465. https://www.mdpi.com/2079-9292/13/13/2465
3. Mahmoud, E. (2025). Enhancing hosting infrastructure management with AI-powered automation. https://www.theseus.fi/handle/10024/882571
4. **Touvron, H., Martin, L., et al.** (2023). LLaMA 2: Open foundation and controlled chat models. *Meta AI Research*. https://ai.meta.com/llama
5. **Osterhof, M.** (n.d.). *Cowrie documentation (v2.5.0)*. https://cowrie.readthedocs.io/
6. **Hindy, H., Bayne, E., Atkinson, R., Tachtatzis, C., & Andonovic, I.** (2020). Network threat classification and the impact of current data sets on intrusion detection systems. *IEEE Access, 8*, 104650–104675. https://doi.org/10.1109/ACCESS.2020.2994769
7. **Sommer, R., & Paxson, V.** (2014). Flow-based intrusion detection: Technologies and challenges. *Passive and Active Measurement Conference (PAM)*. https://link.springer.com/chapter/10.1007/978-3-319-04918-2_17
8. **Deshmukh, S., Rade, R., & Kazi, D. F.** (2019). Attacker forms random profiling of hidden Markov models. *International Journal of Scientific Research and Review*. https://ijsrr.org/down_3848.php
9. **Sladdic, M., Valeros, F., Catania, C., & Garcia, S.** (2023). Master of the crust: Generative honeypots. *Proceedings of the 2023 ACM Workshop on Artificial Intelligence and Security (AISec)*. https://dl.acm.org/doi/10.1145/3605767.3620616