Research Honeypot using Deception Mechanism

Prasanth Bala
221FA19029
Advanced.Computer Science Engineering
Vignan's Foundation for Science Technology and Research
Guntur, India
prasanthchowdarybala@gmail.com

Ajay Potla
221FA19037
Advanced.Computer Science Engineering
Vignan's Foundation for Science Technology and Research
Guntur, India
chowdarya972@gmail.com

Shaik Sohail Ahammed
221FA19051
Advanced.Computer Science Engineering
Vignan's Foundation for Science Technology and Research
Guntur, India
sohailsha2004@gmail.com

Dr.Benson Mansingh P.M.
Senior Assistant Professor
Advanced.Computer Science Engineering
Vignan's Foundation for Science Technology and Research
Guntur, India
benyuva@gmail.com

***Abstract*—The necessity for proactive security measures has been highlighted by the swift evolution of cyberthreats. Honeypots have become powerful instruments that trick attackers into confined spaces so that businesses may examine their methods and plans. The various kinds of honeypots, their function in deception-based security, and their smooth integration into contemporary security frameworks, such as Security Information and Event Management (SIEM) systems, are all covered in this paper. We cover ethical issues, regulatory compliance, and the future of honeypot technology while presenting real-world case studies on SSH, command-line, and web-based honeypots to highlight their efficacy. All things considered, we stress their increasing significance in strengthening organizational security and improving threat intelligence.**

I. Introduction

Advanced security measures are becoming more and more necessary as cyber threats continue to change. As decoys, honeypots draw in attackers and make it easier for security personnel to keep an eye on unwanted activity. Honeypots give attackers the opportunity to reveal their methods and strategies by establishing controlled environments. This study looks at several kinds of honeypots, their real-world uses, and how they might be included into modern security frameworks like threat intelligence systems and SIEM.

II. The Role of Deception in Cybersecurity

Deception plays a critical role in cybersecurity by misleading attackers, diverting their efforts, and exposing their tactics. In honeypots, cyber deception tactics include:

- **Fake Systems and Services:** Constructing virtual environments that give the impression that they are authentic to attackers.
- **Dynamic Identity Spoofing:** Increasing engagement by imitating several systems.
- **Data poisoning:** presenting false information to trick attackers.
- **Behavioral Traps:** In order to fool attackers, AI is used to create realistic system activity.
- **Redirection Strategies:** directing attackers to controlled honeypot environments instead of real production environments.
- **Automated Response Systems:** Using AI-powered defenses that modify deception tactics in response to real-time attack trends.

III. Advanced Honeypot Architectures

Depending on their intended use and integration with an organization's security infrastructure, honeypots can be designed in a variety of ways. Important architectures consist of:

- **Single-Layered Honeypots**: Simple honeypots that mimic a single system or service are known as single-layered honeypots.
- **Multi-Layered Honeypots:** intricate settings that resemble business networks and give intruders multiple points of contact.
- **Honeynets:** Systems of linked honeypots that replicate an entire IT architecture.
- **AI-Enhanced Honeypots:** These employ predictive analytics and machine learning to enhance deception and adjust to emerging threats.
- **Cloud-based honeypots:** These are set up in virtualized settings to keep an eye on and lessen risks to cloud infrastructure.

IV. What is a Research Honeypot?

An purposely weak system or network resource that is set up to entice cybercriminals is called a research honeypot. These honeypots are only there to research malicious activity and attack trends; they don't assist actual users or serve any other justifiable objectives. These systems give researchers information about new dangers, weaknesses, and attacker techniques that they might utilize to create stronger defenses.

V. Objectives of a Research Honeypot

- **Threat Intelligence Gathering:** Gathering information on emerging malware, exploits, and attack trends. By observing the tactics, methods, and procedures (TTPs) that hackers employ, one can gain an understanding of attacker behavior.
- **Creating Security Measures:** Enhancing response plans and intrusion detection systems (IDS).
- **Testing and Assessing Security Tools:** Verifying firewalls, intrusion detection systems, and additional security measures.
- **Academic and Industrial Research:** supplying cybersecurity organizations and research institutions with useful data.

VI. Implementation of Honeypots with Deception Mechanisms

A. Configuring the Environment

- Download and install Ubuntu 22.04 from the Store.
- Set up directories and user credentials.
- Install the essential libraries, Flask, Paramiko, and Python, among other dependencies.

B. Setting Up an SSH Honeypot with Features for Deception

- Use Paramiko to write an SSH honeypot script.
- Use the following to generate SSH keys:

```
ssh-keygen -t rsa -b 2048 -f server.key
```

- To keep an eye on automated attack attempts, implement credential stuffing detection.
- To deceive attackers, insert fictitious accounts and passwords into authentication records.
- Send attackers to remote locations with comparable security flaws.

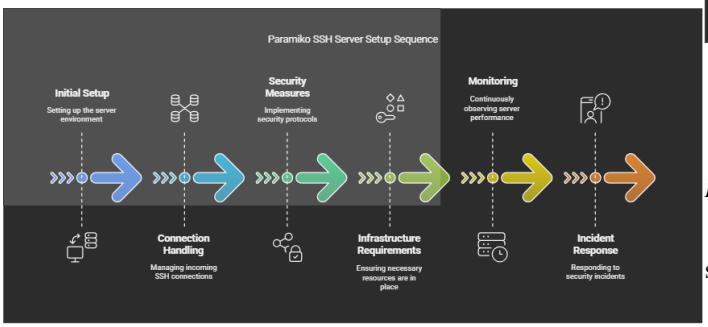

Fig. 1. Paramiko SSH server setup sequence

VII. SSH Honeypot Security Monitoring System

Secure remote access is made possible by the Secure Shell (SSH) protocol, but it is also a major target for online dangers including brute-force attacks and illegal access. The SSH Honeypot Security Monitoring System's disciplined architecture, strong infrastructure, and thorough monitoring workflow are all intended to improve security.

A. *Infrastructure of the System*

Multiple layers make up the system's operation, which guarantees security and effective monitoring:

- **Security Modules:** Implement regulations, identify irregularities, and address dangers.
- **Server Infrastructure:** In order to draw in attackers, server infrastructure replicates vulnerabilities and maintains SSH connections.
- **Database Connection:** Holds user activity, authentication information, and logs for analysis.
- **Back-end:** Manages data processing and session monitoring while connecting infrastructure with user-facing elements.
- **Front-end:** Offers an interface via which administrators may keep an eye on logs, establish policies, and address risks.

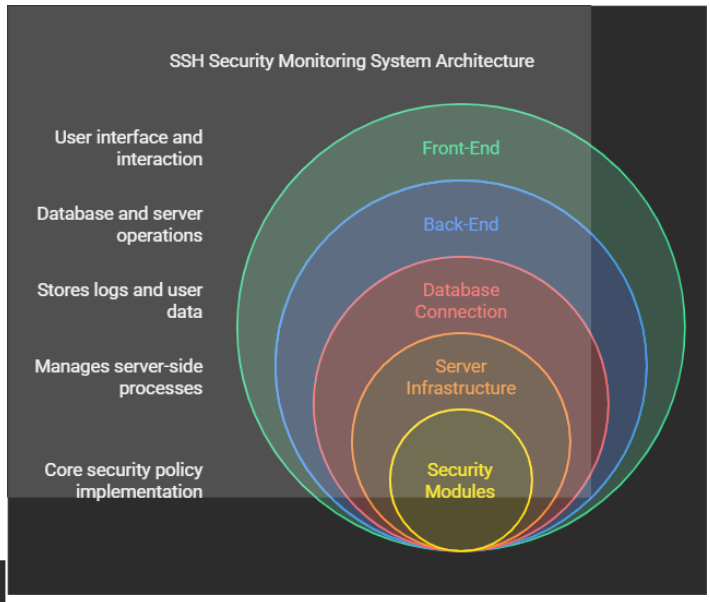

Fig. 2. SSH system architecture

B. *Architecture of the System*

The system is made up of interconnected layers that provide seamless operation:

- **User Interface and Interaction:** An online dashboard for alerts, notifications, and real-time monitoring.
- **Database and Server Operations:** Records SSH activity, such as session commands and authentication attempts, to document the actions of attackers.
- **Implementation of Core Security Policy:** Specifies encryption standards, access control, and threat detection behavioral monitoring.

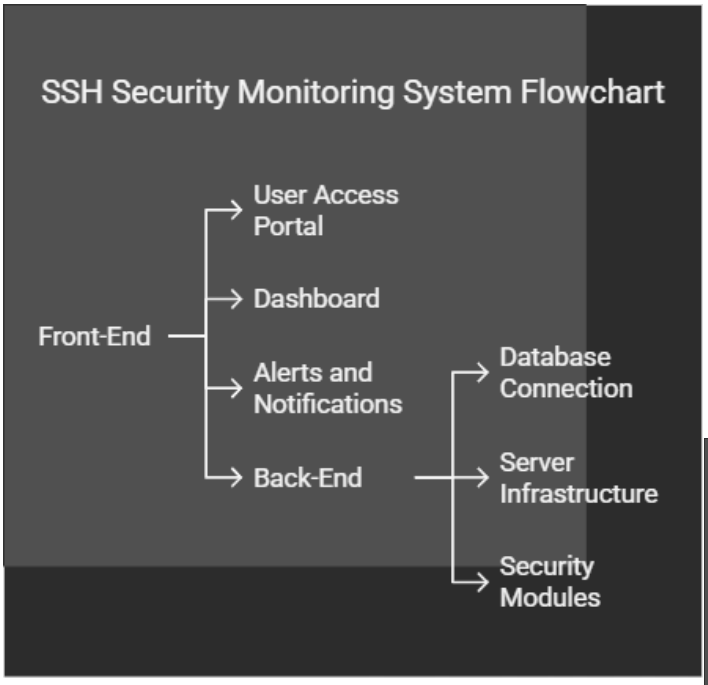

Fig. 3. SSH Architecture

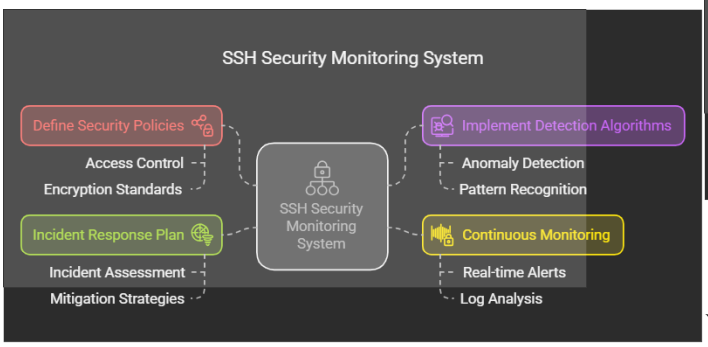

Fig. 4. SSH Architecture

C. *Workflow for the System*

The system detects, logs, and reacts to SSH threats using a systematic workflow:

1) Configuring and Observing:

- To replicate an authentic SSH environment, install the Paramiko SSH Honeypot Server.
- Check for SSH connections and confirm their legitimacy.

2) 2. Verification and Authentication:

- To stop malicious requests, run IP blacklist checks.
- Check for key-based authentication and passwords.
- To examine the behavior of attackers, simulate authentication failures.

3) 3. Management of Sessions:

- To monitor command execution in a honeypot shell, simulate a session.
- For analysis, record and preserve user commands.

4) 4. Behavior Tracking and Warnings:

- Keep an eye out for any unusual trends in the attacker's behavior.
- Inform administrators of any unusual activity.
- Determine typical attack methods for upcoming security upgrades.

5) 5. Mechanism for Incident Response:

- When abnormalities are found, initiate an incident response strategy.
- Log security incidents and end questionable sessions.
- Give forensic details about the sources and techniques of the attack.

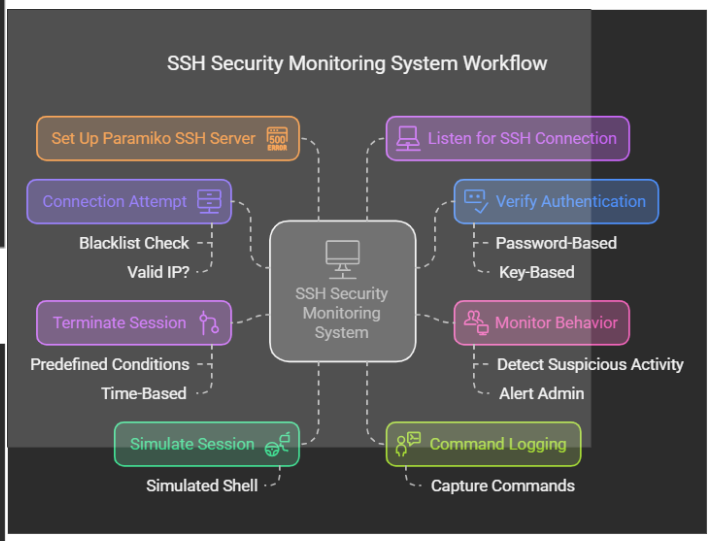

Fig. 5. SSH server workflow

VIII. Implementation of Honeypots with Deception Mechanisms : Setting Up the Environment

A. *Honeypot Installation and Setup Guide*

1) Step 1: Install Ubuntu 22.04 on Windows:

1) Go to Ubuntu 22.04 LTS on the Microsoft Store.
2) After selecting Install, watch for the download to finish.
3) From the Start menu, launch the Ubuntu application after installation.
4) To finish the installation, adhere to the on-screen directions.
5) After installation, restart your computer.

Step 2: Install Ubuntu (Login for the First Time)

```
wsl
sudo apt update && sudo apt upgrade -y
sudo apt install python3 python3-pip -y
python3 --version
```

Step 3: Connect VS Code and PowerShell to Ubuntu

```
wsl --install
mkdir honeypot && cd honeypot
touch ssh_honeypot.py
```

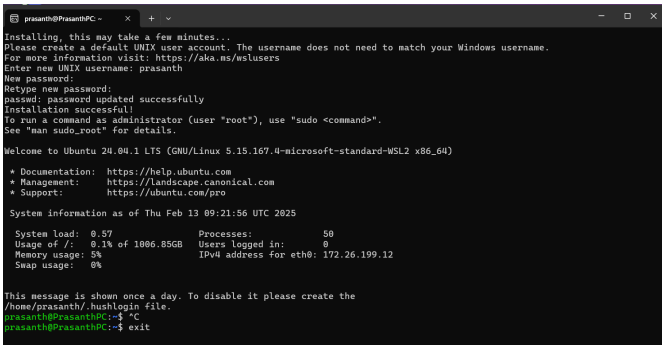

Fig. 6.

```
honeypy.py
web_honeypot.py
wp-admin.html
```

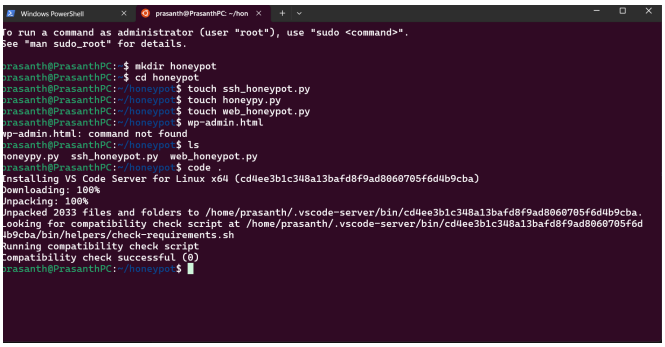

Fig. 7.

B. *Setting Up Requirements*

```
sudo apt update && sudo apt upgrade -y
sudo apt install git python3 python3-pip
sudo snap install code --classic
code --version
```

IX. Case Studies

A. *Case Study 1: SSH Honeypot*

SSH Honeypot Implementation with Paramiko

```
---->Installation of Dependencies
pip install paramiko

---> SSH Honeypot Script (ssh_honeypot.py)

import paramiko
import socket
import threading
import logging

logging.basicConfig(filename=
        "ssh_honeypot.log",
level=logging.INFO,
format="%(asctime)s - %(message)s")

FAKE_USERS = { "admin": "admin123", "root":
"toor", "user": "password" }

class SSHHoneypot(paramiko.ServerInterface):
    def check_auth_password(self, username,
    password):
  if username in FAKE_USERS and FAKE_USERS
        [username] == password:
        logging.info(f"Successful login
attempt: {username} | {password}")
        return paramiko.AUTH_SUCCESSFUL
        else:
            logging.info(f"Failed login
attempt: {username} | {password}")
            return paramiko.AUTH_FAILED
def start_honeypot():
    host_key = paramiko.RSAKey.
    generate(2048)
    server = socket.socket(socket.AF_INET,
    socket.SOCK_STREAM)
    server.bind(("0.0.0.0", 2222))
    server.listen(100)
    logging.info("SSH Honeypot
is running on port 2222")
    while True:
        client, addr =
        server.accept()
        logging.info(f"Connection
received from {addr}")
if __name__ == "__main__":
    start_honeypot()
```

Running the Honeypot

```
----> Start the SSH Honeypot
python3 ssh_honeypot.py
----> Connect to the Honeypot
ssh -p 2222 user@localhost
```

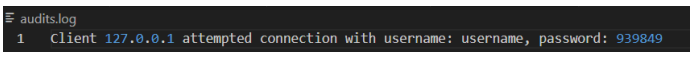

Fig. 8. cmd-audits.log

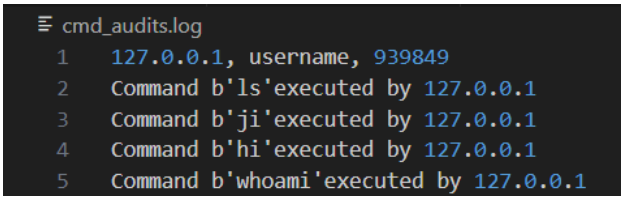

Fig. 9. cmd-audits.log

B. Case Study 2: Argparse Honeypot

```
import argparse
import logging

logging.basicConfig(filename="audit.log",
    level=logging.INFO, format="%(asctime)s
■■■-■%(message)s")
parser = argparse.ArgumentParser
          (description="Fake■CLI■Tool")
parser.add_argument("-a", "--address",
          help="Target■IP■address")
parser.add_argument("-p", "--port",
                  help="Target■Port")
parser.add_argument("--ssh", action="store
■■■■■_true", help="Simulated■SSH■attack")
args = parser.parse_args()
logging.info(f"Attacker■Input:■{args}")
print("Command■not■recognized.■Try■again.")
```

Running the Honeypot

```
----> Start the Honeypot
python3 honeypy.py -a 127.0.0.1 -p 2223 --ssh
 -a 127.0.0.1-> Binds the honeypot to localhost.
 -p 2223 -> Listens on port 2223
  instead of the default SSH port (22).
 --ssh -> Enables SSH honeypot functionality
----> View Logged Activity
cat audit.log
```

```
Client 127.0.0.1 attempted connection with username: username, password: hfhsgdfhs
```

Fig. 10. audits.log

```
127.0.0.1, username, hfhsgdfhs
Command b'hi'executed by 127.0.0.1
Command b'hello'executed by 127.0.0.1
Command b'whoa,\x7fmi\\\x7f\x7f\x7f\x7f\x7f\x7f\x7f\x7f\x7f'executed by 127.0.0.1
Command b'whoami'executed by 127.0.0.1
Command b'ls'executed by 127.0.0.1
```

Fig. 11. cmd-audits.log

C. Case Study 3: Web-Based Honeypot

```
from flask import Flask, request
import logging

app = Flask(__name__)
logging.basicConfig(filename="http_logs.log",
    level=logging.INFO, format="%(asctime)s
■■■■■-■%(message)s")

@app.route("/login", methods=["POST"])
def login():
    data = request.form
logging.info(f"Captured■Credentials:■{data}")
    return "Invalid■credentials", 401

if __name__ == "__main__":
    app.run(port=8080, debug=True)
```

Running the Honeypot

```
----> Start the Web Honeypot
python3 web_honeypot.py
----> Simulate an Attack
curl -X POST -d "username=attacker&password=
■12345" http://127.0.0.1:8080/login
----> View the Captured Request
cat http_logs.log
```

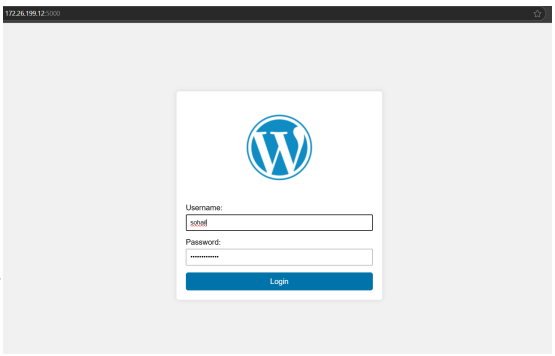

Fig. 12. web page

```
http_audits.log
1  2025-02-13 10:25:14,636 Client with IP Address: 172.26.192.1 entered
2   Username: sohail, Password: hshfksdfjkshu
3
```

Fig. 13. http-audits.log

```
* Serving Flask app 'web_honeypot'
* Debug mode: on
ARNING: This is a development server.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.26.199.12:5000
ress CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 806-795-255
```

Fig. 14. honeypot links

X. Results and Analysis

Each honeypot implementation's data collection yielded insightful information:

- **SSH Logs:** Several unsuccessful attempts at unauthorized login were noted.

```
audits.log
1   Client 127.0.0.1 attempted connection with username: username, password: 939849
```

Fig. 15. cmd-audits.log

```
cmd_audits.log
1   127.0.0.1, username, 939849
2   Command b'ls'executed by 127.0.0.1
3   Command b'ji'executed by 127.0.0.1
4   Command b'hi'executed by 127.0.0.1
5   Command b'whoami'executed by 127.0.0.1
```

Fig. 16. cmd-audits.log

- **Argparse Honeypot:** Recorded a variety of attackers' command inputs.

```
Client 127.0.0.1 attempted connection with username: username, password: hfhsgdfhs
```

Fig. 17. audits.log

```
127.0.0.1, username, hfhsgdfhs
Command b'hi'executed by 127.0.0.1
Command b'hello'executed by 127.0.0.1
Command b'whoa,\x7fmi\\\x7f\x7f\x7f\x7f\x7f\x7f\x7f\x7f\x7f\x7f'executed by 127.0.0.1
Command b'whoami'executed by 127.0.0.1
Command b'ls'executed by 127.0.0.1
```

Fig. 18. cmd-audits.log

- **Web honeypot:** Recorded several phony login attempts.

```
http_audits.log
1   2025-02-13 10:25:14,636 Client with IP Address: 172.26.192.1 entered
2   Username: sohail, Password: hshfksdfjkshu
3
```

http-audits.log

XI. Ethical and Legal Considerations

Although honeypots offer significant security advantages, they can present moral and legal dilemmas:

- **Privacy Concerns:** Making sure that user privacy rights are not violated by collected data.
- **Deterrence vs. Entrapment:** balancing legal frameworks to prevent charges of illegal entrapment.
- **Regulatory Compliance:** Complying with international security laws like HIPAA and GDPR.
- **Ethical Consequences of Deceitful Security Practices:** Analyzing the effects of deceptive security on user confidence and moral obligation

XII. Future Trends in Deceptive Honeypots

As technology advances, honeypots continue to change. Future studies could concentrate on:

- **Honeypots with Quantum Security**
  - Utilize **post-quantum cryptographic techniques:** Employ post-quantum cryptography strategies to fend off quantum computer threats.
  - Implement **lattice-based encryption and quantum key distribution:** For increased security, use quantum key distribution and lattice-based encryption.
- **AI-Powered Automated Response Systems**
  - Integrate **machine learning:** These systems use machine learning to instantly assess assault patterns
  - Enable **dynamic deception strategies:** Make it possible for dynamic deception tactics to change according to the actions of the attacker.
  - Use AI-driven **predictive security:** Predictive security powered by AI can be used to stop threats before they get worse.
- **Blockchain for Data Integrity**
  - Decentralized, tamper-proof ledgers with secure honeypot logs.
  - Assure reliable threat intelligence sharing and data immutability.
  - Stop hackers from altering logs to hide their activities.
- **Deception-Oriented Active Defense Strategies**
  - Simulate **realistic user behavior** To trick attackers, mimic real-world user behavior and AI-generated system activity.
  - Use automated redirection strategies to reroute attackers away from important resources.
  - Create **multi-layered deception environments** that change according to the types of attacks.
- **Honeypots as Proactive Security Tools**
  - Make the switch to active threat counteraction from passive monitoring. By confusing and delaying cybercriminals' attempts, you can lessen their effectiveness.
  - Integrate deception into larger security systems to improve cybersecurity generally.

XIII. Conclusion

Due to their ability to effectively detect, analyze, and mitigate cyber threats, honeypots are an essential part of contemporary cyber-security methods. Through deception techniques including behavioral traps, artificial intelligence (AI)-driven response systems, and phony credentials, honeypots give security professionals vital information about new attack routes. Our case studies show how SSH, command-line, and web-based honeypots are used in practice and show how well they catch malicious activity and illegal access attempts.To guarantee adherence to international security laws, the moral

and legal issues underlying honeypot deployment must be properly handled. The capabilities of honeypots are anticipated to be substantially improved by upcoming developments in AI, blockchain, and quantum computing, making them a crucial cybersecurity tool. In the end, strategically placing honeypots can greatly enhance threat detection, fortify defenses, and help create a more robust cybersecurity architecture.

REFERENCES

[1] Spitzner, L. (2003). Honeypots: Tracking Hackers. Addison-Wesley.
[2] Provos, N., Holz, T. (2007). Virtual Honeypots: From Botnet Tracking to Intrusion Detection. Addison-Wesley.
[3] Joshi, R. C., Sardana, A. (2011). Honeypots: A New Paradigm to Information Security. Springer.
[4] Mokube, I., Adams, M. (2007). Honeypots: Concepts, Approaches, and Challenges. Proceedings of the 45th ACM Southeast Conference.
[5] Nawrocki, M., et al. (2016). A Survey on Honeypot Software and Research.
[6] Alata, E., et al. (2006). Internet Attacks Monitoring with a New Generation Honeypot.
[7] Seifert, C., et al. (2007). Capture-HPC: High-Interaction Honeypot for Malware Collection.
[8] Krawetz, N. (2004). Introduction to Honeypots. ACM Journal of Network Security.
[9] Rajab, M. A., et al. (2007). A Study of Spamming Botnets using Honeypots.
[10] Zeltser, L. (2014). Practical Malware Analysis Using Honeypots.
[11] Pouget, F., Dacier, M. (2004). Honeypot-Based Forensics.
[12] Harrop, W., Armitage, G. (2006). Defining and Evaluating Honeypot Technologies.
[13] Bowen, B. M., et al. (2009). Baiting Inside Attackers Using Decoy Documents.
[14] Krawetz, N. (2004). Anti-Honeypot Technology. BlackHat Conference.
[15] Spitzner, L. (2004). The Value of Honeypots in Threat Intelligence.