Enhancing False Positive Detection in IDS/IPS Using Honeypots: A Case Study with CSE-CIC-2018 Dataset

A thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Computer Science and Engineering
by Research

by

CHOWDURU RAMACHANDRA SHARMA
201302177
chowduru.ramachandra@research.iiit.ac.in

Advisor: Prof. SHATRUNJAY RAWAT

International Institute of Information Technology Hyderabad
500 032, India

May 2024

International Institute of Information Technology Hyderabad
Hyderabad, India

CERTIFICATE

This is to certify that work presented in this thesis proposal titled ***Title of the research thesis*** by *Name of the author* has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Advisor: Prof. SHATRUNJAY RAWAT

Abstract

Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) protect computer networks against unauthorised access and malicious traffic. False positives, where legitimate network traffic is incorrectly flagged as malicious, compromise the efficiency of these systems. False positives waste valuable resources and reduce the overall reliability of IDS/IPS alerts, leading to potential security risks.

Our work aims to address the issue of false positive detection in IDS/IPS by utilising the capabilities of honeypots. Honeypots are deceptive systems designed to mimic legitimate services and attract potential attackers, providing a controlled environment to analyse their behaviour without posing any risk to the network infrastructure.

This thesis explores the evolution of honeypots, from their beginning as passive research tools to their modern applications as refined deception mechanisms. The diverse types of honeypots are examined, indicating their potential to lure attackers and provide valuable insights into malicious activities without risking network assets.

Next, the research analyses an in-depth survey of existing false positive mitigation techniques in IDS/IPS, analysing their strengths and limitations. Next, various honeypots are studied to identify their potential to effectively capture and differentiate legitimate and malicious traffic. The thesis proposes a hybrid approach that integrates the information gathered from honeypots with the output of the IDS/IPS, enabling more intelligent and precise decision-making for false positive detection.

A comprehensive experimental setup is designed, simulating real-world network scenarios to evaluate the effectiveness of the proposed solution. The CSE-CIC-2018 dataset, a widely used and realistic cybersecurity dataset, is employed for experimentation to assess the effectiveness of our approach. Quantitative metrics such as true positive rate, false positive rate, and accuracy are measured to gauge the improvement achieved by the hybrid system.

When honeypots are incorporated into the IDS/IPS framework, the results significantly reduce false positives. Furthermore, the study reveals insights into attackers' behaviour, aiding in developing more robust security policies and threat intelligence.

In conclusion, this thesis presents a novel approach using honeypots to mitigate false positive detections in IDS/IPS. The integration of honeypots supports the network to make informed decisions, improving the overall efficiency and reliability of intrusion detection and prevention systems. By reducing false positives, networks can allocate resources more effectively and respond to genuine security threats.

Contents

Chapter Page

1 Introduction . . . 1
1.1 Motivation . . . 1
1.2 Research Scope . . . 2
1.3 Thesis Organization . . . 2

2 Background . . . 3
2.1 Honeypots and Honeynet . . . 3
2.1.1 Honeypots . . . 3
2.1.2 Types of Honeypots . . . 4
2.1.2.1 Low Interaction Honeypots(LIH): . . . 4
2.1.2.2 Medium Interaction Honeypots: . . . 4
2.1.2.3 High Interaction Honeypots(HIH): . . . 4
2.1.3 Evolution of Honeypots . . . 4
2.1.4 Honeynet . . . 5
2.1.5 Evolution of Honeynets . . . 6
2.1.5.1 GenI Honeynet Architecture: . . . 6
2.1.5.2 GenII Honeynet Architecture: . . . 7
2.1.5.3 Virtual Honeynets: . . . 9
2.1.5.4 Distributed Honeynets: . . . 9
2.1.6 Risk of Honeypot and Honeynet . . . 10
2.2 Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) . . . 11
2.2.1 Intrusion Detection Systems (IDS) . . . 11
2.2.2 Intrusion Prevention System (IPS) . . . 11
2.2.3 Evolution of IDS/IPS . . . 12

3 Related Work . . . 13
3.1 Related papers . . . 13
3.1.1 Honeyd: . . . 13
3.1.2 Niels Provos: . . . 14
3.1.3 Collapsar . . . 15
3.1.3.1 Router-Based Redirection . . . 15
3.1.3.2 End System-Based Redirection . . . 16
3.1.3.3 Collapsar Center Functionality . . . 16
3.1.3.4 Dynamic Creation of HIHs . . . 16
3.1.3.5 Gateway for Intended Traffic . . . 16
3.1.3.6 Overloading Concerns . . . 17
3.1.3.7 Delay Mitigation . . . 17
3.1.3.8 Limitations . . . 17
3.1.4 Honeybrid: . . . 17
3.1.5 Ip fingerprinting: . . . 19

3.1.5.1 Latency and Detectability . . . 19
3.1.5.2 Dynamic Honeypot Generation . . . 19
3.1.5.3 KVM-Based Spin-Up . . . 19
3.1.5.4 Honeyd for LIHs Generation . . . 19
3.1.5.5 VMX for HIHs and Virtual Networks . . . 19
3.1.5.6 VNX for Scenario Reconfiguration . . . 19
3.1.5.7 LXC-Based VNX . . . 20
3.1.5.8 Customizable Standards . . . 20
3.1.5.9 Conclusion . . . 20
3.1.6 Potemkin Virtual Honeyfarm: . . . 20
3.1.6.1 Architecture . . . 20
3.1.6.2 Packet Handling and Containment: . . . 21
3.1.6.3 Universe Concept for Containment . . . 21
3.1.6.4 Scalability Optimization . . . 21
3.1.6.5 Conclusion . . . 22
3.1.7 Anomaloy detectors: . . . 22
3.1.7.1 Honeypot Coupling . . . 22
3.1.7.2 Anomaly Detection Techniques . . . 22
3.1.7.3 Detection of Server and Client Attacks . . . 23
3.1.7.4 Performance of Shadow Honeypots . . . 23
3.1.7.5 Drawbacks and Considerations . . . 23
3.1.8 Shark: Spy Honeypot with Advanced Redirection Kit . . . 24
3.1.8.1 Introduction: . . . 24
3.1.8.2 Key Features: . . . 24
3.1.8.3 Implementation: . . . 24
3.1.8.4 Dynamic Redirection : . . . 24
3.1.8.5 Benefits: . . . 25
3.1.8.6 Conclusion; . . . 26
3.2 Potential Risks and Challenges of Honeypots and IDS/IPS Systems . . . 26
3.2.1 Risks: . . . 26
3.2.2 Challenges: . . . 26
3.3 Honeycomb: . . . 27
3.4 T-POT CE (Telekom Security's Honeypot Platform Community Edition) . . . 27
3.4.1 Architecture . . . 27
3.4.2 Honeypot Categories . . . 28
3.4.3 Resilience and Responsiveness . . . 29
3.4.4 Conclusion . . . 29

4 Design Architecture . . . 30
4.1 Architecture: . . . 30
4.1.1 Components: . . . 30
4.1.1.1 IDS/IPS: . . . 30
4.1.1.2 Honeybrid: . . . 30
4.1.1.3 Honeynet: . . . 31
4.2 Data Control: . . . 31
4.3 Data Capture: . . . 32
4.4 Advantages Over Other Architectures: . . . 32
4.4.1 Over Collapsar: . . . 32
4.4.2 Over Anomaly-Based Honeypots: . . . 33
4.4.3 Over Potemkin: . . . 33
4.4.4 Over Shark: . . . 33

4.4.5 Over Dynamic Hybrid Honeypot System Based on Transparent Traffic Redirection Mechanism: . . . 33
4.5 Advantages: . . . 34
4.6 Limitations: . . . 34

5 Implementation . . . 35
5.1 Components: . . . 35
5.1.1 Apache server: . . . 35
5.1.2 Gateway: . . . 35
5.1.3 Snort (NIDS): . . . 36
5.1.3.1 Alerting mechanism: . . . 37
5.1.4 Honeybrid: . . . 37
5.1.5 Configuring Netfilter rules: . . . 37
5.1.5.1 Mod_alert Module: . . . 38
5.1.5.2 Target Rule: . . . 38
5.1.5.3 Flexible Redirection: . . . 38
5.1.5.4 Rate-Limiting and Control Rule Implementation: . . . 39
5.1.6 TPOTCE(HONEYNET): . . . 39
5.1.6.1 Configuring Snare and Tanner in TPOT CE: . . . 40

6 Experimental Evaluation . . . 42
6.1 Evaluation through dataset: . . . 42
6.1.1 Tool generated traffic: . . . 42
6.1.1.1 True Positive Simulation: . . . 42
6.1.1.2 Dataset Generation and Evaluation: . . . 42
6.1.1.3 Evaluation of Snare using synthetic data: . . . 43
6.1.1.4 False Positive Simulation: . . . 43
6.1.2 CSE-CIC-IDS2018 dataset: . . . 44
6.1.2.1 Modifying Dataset for Architecture: . . . 45
6.1.2.2 CSE CIC Dataset Challenges: . . . 45
6.1.2.3 Evaluation of CSE CIC Dataset: . . . 45
6.2 Performance: . . . 48
6.2.1 IDS/IPS Performace: . . . 48
6.2.2 Honeybrid Performace: . . . 48

7 Conclusion . . . 50
7.1 Thesis Summary: . . . 50
7.2 Future Work: . . . 50

Bibliography . . . 52

List of Figures

Figure Page

2.1 GenI Architecture 6
2.2 GenII Architecture 8

3.1 Honeyd Architecture 14
3.2 Redirection sequence diagram 15
3.3 Complete Architecture of Collapsar System 16
3.4 Redirection sequence diagram 18
3.5 Honeyfarm architecture for Potemkin virtual Honeyfarm 21
3.6 Shadow Honeypot architecture. 23
3.7 Mechanism of redirection 25

4.1 Proposed Architecture 32

5.1 Architecture design of Implementation 36
5.2 TPOT CE Honeypots and port configuration 41

6.1 Tanner XSS attack detection 43
6.2 Tanner top attacks 43
6.3 Tanner Dashboard visualising attack data 44
6.4 Nmap scan alerts in TPOT CE Suricata tool 47
6.5 TPOT CE Dashboard 48

List of Tables

Table Page

5.1 Configuration of Apache Server 35
5.2 Configuration of Snort 37
5.3 Configuration of TPOT CE Honeynet 40

6.1 CSE-CIC-IDS 2018 Dataset traffic distribution 45
6.2 Snort alert classification with PulledPork v0.7.4 46
6.3 Protocol traffic and TPOT CE port mapping 46

Symbols

Chapter 1

Introduction

1.1 Motivation

Protecting computer networks against malicious activities is paramount in today's rapidly evolving digital landscape. Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) have emerged as crucial components within modern networks, acting as the first line of defence against potential threats. Their role in identifying and preventing unauthorised access attempts is pivotal to maintaining the integrity and security of network infrastructures.

However, the effectiveness of IDS/IPS systems can be degraded by the persistent challenge of false positives. False positives occur when legitimate network traffic is inaccurately classified as malicious, triggering unnecessary alerts. These false positives lead to a depletion of valuable resources and deteriorate the confidence of network administrators and security personnel's confidence in the alerts' reliability. This critical issue highlights the need for enhanced precision in distinguishing genuine threats from benign activities.

Amidst this backdrop, Honeypots come across as an exciting possibility. These purpose-built systems are crafted to mimic authentic network services and applications, creating an environment that attracts potential attackers. The core objective of honeypots is to redirect and expose malicious activities, offering a controlled environment to gain insights into the methods and behaviours of attackers, all while isolating genuine production systems from harm.

Despite the potential advantages of honeypots, their widespread adoption remains limited for various reasons. Factors such as resource allocation challenges, concerns about generating false positives from honeypot data, and uncertainties around the legal and ethical implications of luring attackers can prevent organisations from fully utilising their capabilities.

Bridging this gap of False positives and advancing the realm of intrusion detection and prevention, this thesis explores a compelling area — the integration of honeypots into the IDS/IPS framework. This innovative collaboration seeks to leverage the strengths of both technologies. By fusing the precision of IDS/IPS systems with the insights gained from honeypots, a comprehensive solution is pictured—one that can significantly mitigate the issue of false positives and amplify the accuracy of threat detection.

In cracking this research journey, we dive into the evolution of honeypots, the complexities of false positive detection, and the underpinning motivations that shape the underutilisation of honeypot technology. Through a rigorous examination and evaluation of the proposed hybrid approach, backed by real-world data from the CSE-CIC-2018 dataset, we aspire to not only shed light on the efficacy of this strategy but also contribute to the collective knowledge in strengthening network security against an evolving landscape of cyber threats.

1.2 Research Scope

The scope of this research encompasses the following key areas:

- Investigation of Honeypot Evolution: Exploration of the historical progression and contemporary applications of honeypots within the cybersecurity landscape.
- Evaluation of Existing False Positive Detection Techniques: Assessment of current methods used to identify and manage false positives in IDS/IPS systems, focusing on their effectiveness and challenges.
- Development of an Integrated Architecture: Designing a hybrid architecture that seamlessly integrates IDS/IPS components with strategically deployed honeypots, allowing for enhanced false positive detection accuracy.
- Utilisation of CSE-CIC-2018 Dataset: The practical implementation and evaluation of the proposed architecture using the CSE-CIC-2018 dataset, a realistic cybersecurity dataset that simulates genuine network traffic.
- Performance Metrics and Analysis: Quantitative measurement of false positive rates, true positive rates, accuracy, and related metrics to assess the effectiveness of the hybrid architecture. Additionally, analysis of insights gained from attacker behaviour observations.

1.3 Thesis Organization

The remainder of this thesis is structured as follows:

- Chapter 2 provides an in-depth exploration of the evolution of honeypots, tracing their historical development and discussing their diverse applications.
- Chapter 3 analyses existing false positive detection techniques employed in IDS/IPS systems, highlighting their strengths, limitations, and associated challenges.
- Chapter 4 details the design and architecture of the proposed hybrid IDS/IPS-honeypot system, outlining its components, communication mechanisms, and data exchange processes.
- Chapter 5 presents the utilisation of the CSE-CIC-2018 dataset for evaluating the performance of the hybrid architecture and provides a comprehensive analysis of the obtained results.
- Chapter 6 delves into the reduction in false positives achieved by the hybrid architecture, presenting quantitative measurements and discussing the broader implications of the findings.
- Chapter 7 discusses insights into attacker behaviour derived from the honeypot-collected data and their potential impact on threat intelligence.
- Chapter 8 summarises the research findings, draws conclusions, and outlines potential avenues for future research.

Chapter 2

Background

As many business and critical services run online, attackers find new ways to exploit vulnerabilities for unauthorised access, data theft, and disruption. IDS and IPS have become crucial because the number and complexity of cyberattacks have grown immensely. More than traditional security measures like firewalls and antivirus software are needed to keep up with the evolving tactics of cybercriminals. IDS and IPS provide an additional layer of defence, helping organisations identify and respond to threats before they can cause significant damage. In a world where data breaches, ransomware, and other cybercrimes are rising, IDS and IPS play a critical role in maintaining the security and integrity of digital systems and networks.

2.1 Honeypots and Honeynet

The ever-evolving landscape of cybersecurity has necessitated the development of innovative techniques to understand and counteract security threats. Among these techniques, the concepts of honeypots and honeynets have emerged as essential tools for gaining insights into malicious activities and adversary behaviours. This chapter delves into the background of honeypots and honeynets, elucidating their significance in enhancing our understanding of security threats. Unlike IDS/IPS or firewalls, which are easier to define as they solve specific problems, Honeypots are in contrast to these technologies. Honeypots are involved in prevention, detection, information gathering, and more. In simple terms, a honeypot is an information system resource whose value lies in the unauthorised or illicit use of that information.

2.1.1 Honeypots

A honeypot is a controlled system or network component deliberately made vulnerable to attract and interact with attackers. It imitates a real system or network but contains no actual valuable information. When attackers target a honeypot, their activities are carefully monitored and recorded. This helps security professionals understand attack methods, learn about new threats, and gather data that can be used to enhance cybersecurity measures. Honeypots are decoys that provide valuable insights into the techniques used by malicious attackers without risking the compromise of actual sensitive data.

A honeypot need not be a physical or virtual system; it can be an emulated service or a digital entity. In the case of emulated services, honeypots can convincingly mimic various network protocols using scripts to respond as actual services would, enticing potential attackers. On the other hand, digital entities, like Honeytokens, have no inherent production value but act as bait to detect malicious activity.

2.1.2 Types of Honeypots

Based on the level of interaction with potential attackers and the associated risks, honeypots can be categorised as low, medium, or high interaction.

2.1.2.1 Low Interaction Honeypots(LIH):

Low-interaction honeypots are relatively simple and less resource-intensive to deploy. They mimic only specific aspects of services or systems, often emulating common vulnerabilities and weaknesses, which makes them easy to deploy. These honeypots are designed to gather basic information about potential threats without exposing the network to significant risks. Since they do not run actual services, they have limited interaction with attackers and are less likely to be compromised. Low-interaction honeypots are typically used for early detection and as an alerting mechanism, providing valuable insights into known attacks and attack patterns.

2.1.2.2 Medium Interaction Honeypots:

Medium interaction honeypots strike a balance between realism and security risk. They simulate more comprehensive services and systems compared to low-interaction honeypots. While they provide a more accurate representation of real systems, they still have limitations regarding their full functionality. Medium-interaction honeypots interact with attackers to a greater extent, allowing for the capture of more detailed attack data and techniques. They are used for a broader range of threat analysis, providing a more in-depth understanding of an attacker's methods and intentions. However, they are less risky than high-interaction honeypots because they are designed to contain potential threats effectively.

2.1.2.3 High Interaction Honeypots(HIH):

High-interaction honeypots are the most complex and realistic of the three categories. They emulate actual systems and services, including real operating systems and applications. These honeypots are highly interactive and engage deeply with potential attackers. As a result, they are the most resource-intensive and carry the highest risk because they run actual software that may contain vulnerabilities. High-interaction honeypots aim to provide a nearly identical environment to a production system, making them extremely attractive to attackers. While they yield the most comprehensive insights into attack techniques and intentions, they require meticulous monitoring and maintenance to minimise security risks. That is why they are risky and very complex to deploy. Real physical systems must be configured to attract attackers and minimise the risk of attackers using Honeypot to attack other people. High-interaction honeypots are typically used in advanced threat research.

Other categorisations of Honeypots are server and client Honeypots. Client honeypots mimic vulnerable client devices and wait to be attacked by malicious servers. In contrast, server honeypots emulate server systems and services to engage actively with incoming connections from potentially malicious clients.

2.1.3 Evolution of Honeypots

The evolution of honeypots has been an exciting journey, driven by the need to understand cyber threats better, enhance cybersecurity strategies, and gather intelligence on malicious activities. Here is a detailed look at the critical stages in the evolution of honeypots. In the early stage, Honeypots first emerged as research projects to study hacker behaviour and attack techniques. In the early days, honeypots were relatively simple and were mainly used by cybersecurity researchers and hobbyists. The focus was learning about hacking methods and documenting

attackers' interactions with these decoy systems. This is where honeypots evolved from real systems to emulated services.

As cyber threats became more advanced, the cybersecurity community recognised the potential of honeypots as proactive defence tools. Security professionals started deploying honeypots in corporate networks to supplement intrusion detection systems (IDS) and gain insights into emerging threats. The focus shifted from pure research to practical application, with honeypots used to identify systems and network vulnerabilities.

Later, the concept of production honeypots gained traction, where organisations integrated honeypots into their security infrastructure. These production honeypots were designed to mimic real systems and services within an organisation's network. They provided early warning signs of attacks and helped security teams understand the tactics and methods employed by attackers.

Honeypots started playing a vital role in threat intelligence. Data collected from honeypots began to be analysed and shared among security professionals to understand global trends, new attack vectors, and evolving attacker behaviours. Integrating honeypot data into threat intelligence platforms enhanced the collective knowledge about cyber threats.

The beginning of cloud computing and virtualisation brought new dimensions to honeypot technology. Cloud-based honeypots and virtual honeynets allowed for scalable and cost-effective deployments. With cloud-based architecture, Honeypots need not be physically present in the network.

2.1.4 Honeynet

Honeypots have evolved beyond single systems to include entire networks called honeynets. A honeynet is a network of interconnected honeypots which evolved as part of information gathering. It is like a larger-scale version of a honeypot that simulates an entire network with various systems, services, and vulnerabilities. They provide whole networks of systems for attackers to interact with, making them more interactive of all honeypots. However, they are also one of the most complex honeypot solutions, involving much work and risk. Honeynets offer a more comprehensive view of attacker behaviours across multiple entry points. They can mimic different types of networks, such as a corporate network, a data centre, or even an Internet of Things (IoT) environment.

Honeynet's architecture aims to create a highly controlled network where you can contain and capture all activity. The two crucial requirements for Honeynet architecture are data control and data capture. Data Control defines how an activity is contained within the Honeynet without the attacker knowing it. Data capture is logging all activity without the attacker knowing it. Data control always takes priority over data capture.

Data control is what mitigates the risk. By risk, we mean there is always the potential for an attacker using Honeynet to attack or harm non-Honeynet systems. More freedom for the attacker meant more data capture and risk to non-Honeynet systems. So, data control and data capture should be balanced according to the network requirement and risk thresholds. One of the best ways to approach data control is not to rely on a single mechanism. Instead, implement data control using layers, such as counting outbound connections, intrusion prevention gateways, or bandwidth restrictions.

As with data control, the best practice for data capture is to use multiple mechanisms of information capture. One of the critical aspects of data capture is to consider encryption, as most of the attacker's activity happens over encryption channels. Honeypots should be tightly coupled with the network systems to minimise the attacker's ability to detect capture. Captured data should not be stored locally in the Honeypot systems as this data can be detected, modified, and deleted by attackers.

The third important aspect of Honeynet is data collection. Data collection in the context of Honeynets refers to systematically gathering and storing the data captured from multiple honeypots or across a honeynet. It involves

aggregating and centralising the data collected from various honeypot instances or network segments within the Honeynet. Data collection enables security analysts to correlate information from different sources, identify attack patterns across the network, and gain a comprehensive view of the threat landscape. This centralised data repository aids in in-depth analysis and aids in making informed decisions about mitigating security risks.

2.1.5 Evolution of Honeynets

In Gen1 architecture, the first generation of honeypots, we have a setup that includes a firewall and an Intrusion Detection System (IDS) to manage and capture data related to cyberattacks. Here is a simplified breakdown of how it works:

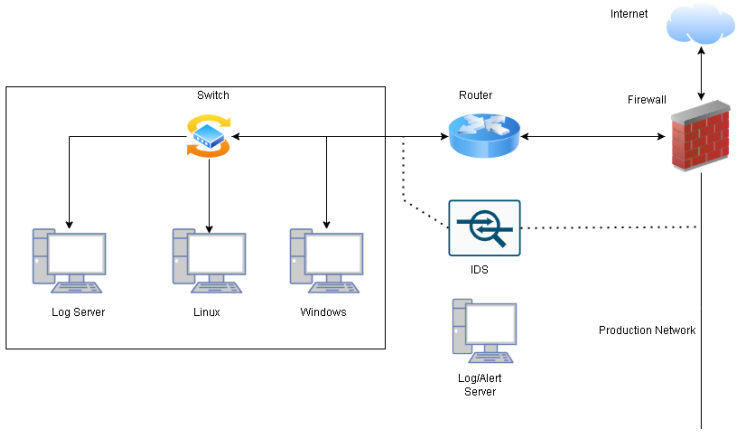

Figure 2.1 GenI Architecture

2.1.5.1 GenI Honeynet Architecture:

Firewall (Gateway):

- The firewall acts like a protective gate for the honeypot network.
- It has three main parts:
  a) External interface (Network): This is where incoming traffic from the internet comes in.
  b) Internal interface (Honeynet): This connects to the honeypot network where the honeypots are.
  c) Management interface: This is used for extracting logs and managing the firewall.
- The firewall is like a security guard that watches over the traffic from the honeypot network.
- It can control the data flow (like managing how much data can pass through) and block certain connections.

IDS (Intrusion Detection System):

- The IDS helps spot unusual or suspicious activities in the network.
- It has two parts:

a) One with an IP address (connected to the production network) is used to manage the IDS and collect data.

b) Another part (connected to the router) without an IP address: This quietly listens to the traffic without participating actively.

- The IDS monitors all the data going to and from the honeypots.
- It collects logs that record network transactions (like who is talking to whom), network traffic (the actual data being sent), what the honeypots are doing, and any alerts it detects.

Data Collection:

- Data collection combines information from various sources like the IDS, firewall, and Network Intrusion Detection System (NIDS).
- It gathers logs and records about network transactions, traffic, what the honeypots are doing, and any alerts from the IDS.

Data Capture Categories:

- There are four main types of data capture:

a) Network transaction recording: Keeps track of who communicates with whom, what protocols they use, and which ports they connect.

b) Network traffic recording: Records the data sent between devices.

c) Host activity recording: Monitors what the honeypots are doing.

d) IDS alerts: Notes any suspicious activity that the IDS detects.

In GenI honeypots, the focus is on low interaction to reduce the risk of exposing vulnerabilities. However, because the IDS mainly looks at network traffic and lacks advanced logging, a clever attacker might use encryption or other tricks to hide their activities from the IDS.

2.1.5.2 GenII Honeynet Architecture:

In the Gen2 honeypot architecture, a significant enhancement is the introduction of the Honeywall gateway. The Honeywall is a game-changer in Honeynet deployment and operation, significantly simplifying the process compared to Gen1 architecture. In Gen1, three separate components handled various functions like data control and capture, making the setup more complex.

This central component is pivotal in data control and capture, revolutionising Honeynets' operations. Here is a detailed breakdown of Gen 2 components.

Honeywall Gateway:

- The Honeywall is the cornerstone of Gen2 honeypots, serving as a centralised control and data capture point.
- It streamlines the deployment of Honeynets, simplifying the process compared to Gen1 architecture.
- This gateway collects and stores crucial data generated by Honeypots, explicitly focusing on capturing the attacker's keystrokes and system application logs.

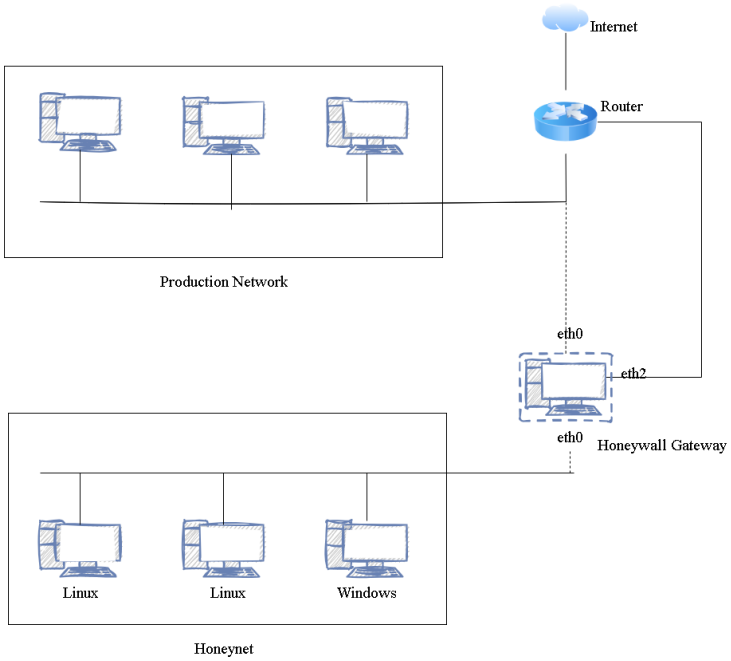

Figure 2.2 GenII Architecture

Sebek Integration:

- Gen2 honeypots leverage Sebek, a critical component for capturing intruder keystrokes and gaining insight into the impact of attacks.
- Sebek logs originating from each Honeypot are consolidated and stored within the Honeywall server.

Data Control:

- The Honeywall incorporates various data control modes, enhancing security and management:
  - Connection rate limiting: This mode limits outgoing connections, helping prevent excessive data exfiltration.
  - Packet drop: Malicious packets are dropped using Snort-Inline, bolstering the network's defence.
  - Packet replacement: Snort-Inline facilitates packet replacement, ensuring that potentially harmful packets are substituted with safe alternatives.

Data Capture:

- As mentioned, data capture contains more information than Gen1, Sebek data from Honeypot.

Versatile Functionality:

- The Honeynet gateway is a versatile device, serving multiple roles simultaneously:
  - Data control device: It manages the data flow within the Honeynet, imposing restrictions as needed.

- Data capture device: The Honeywall collects and stores valuable data, providing insights into malicious activities.
- Data storage device: It is a repository for Honeypot-generated logs and Sebek data.

Management Interface:

- An additional network interface is dedicated to management, enhancing security by segregating Honeynet traffic from Honeywall management traffic.
- This secure remote management interface ensures administrators can oversee and control the Honeynet environment without exposing it to threats.

GenII honeypot architecture introduces the Honeywall gateway, a revolutionary component that centralises data control and capture. It simplifies Honeynet deployment and management, provides extensive data insights through Sebek integration, and offers robust data control mechanisms. With the Honeywall, Gen2 honeypots take a giant leap forward regarding security and effectiveness in countering cyber threats.

2.1.5.3 Virtual Honeynets:

The concept of virtual honeypots revolves around consolidating all aspects of a honeynet onto a single computer through virtualisation technology. This approach minimises costs and streamlines data capture management, making it an attractive option for small- and large-scale deployments.

Router-level virtualisation can be implemented with GNS3 software to imprint a network. Some separate the gateway in one physical machine, and all honeypots are virtualised in another physical device, which is more secure and flexible to add extra data control and data capture methods. It is easy to manage Honeypot, like cloning and starting suspended instances.

Its limitations are increased risks of fingerprinting in a virtual environment. Virtual machines can be identified by active and passive OS fingerprinting, banners (of services like SSH), etc. Running multiple VMs on a single physical host can consume significant resources. This may lead to performance issues and impact the accuracy of the Honeypot's emulation. So, while virtual honeypots are a powerful tool in our cybersecurity toolbox, they come with their challenges. Balancing the benefits and limitations is critical to making the most of this technology in the ever-evolving world of cyber defence.

2.1.5.4 Distributed Honeynets:

Distributed honeypots are a strategic advancement in the world of cybersecurity. They enable the capture of malicious activities across multiple networks, providing a more comprehensive and effective means of threat detection.

Individually Distributed Honeynets with Centralized Data Collection

One approach to distributed honeynets involves deploying individual honeypots across different networks while centralising data collection. This is often achieved using tunnelling techniques. Each network hosts its honeypots, and the data they collect is securely funnelled to a central repository. This architecture simplifies deployment and allows for flexibility in relocating honeypots without physically moving them. However, it introduces latency due to the tunnelling process and may be susceptible to detection by an astute attacker.

Central Honeynet Farm with VPN Tunneling

A more centralised approach establishes a honeynet farm as a central hub for traffic routed via VPN tunnels. Distributed hosts or routers can be configured to route traffic to this central honeynet farm using various techniques, such as GRE Tunneling. This architecture is relatively easy to set up and maintain, ensuring all traffic is funnelled through the central Honeynet for analysis. However, the drawback lies in the potential for increased latency, primarily due to the tunnelling process and gateway configurations. Latency can tip off an attacker that they are interacting with a honeypot.

Policy-Based Routing and Packet Mangling

Policy-based routing can be implemented for each Honeypot, utilising tunnel interfaces to address the issue of latency and detection. Source IP addresses are crucial in routing traffic appropriately. Packet mangling is employed to blur the network topology and make it more challenging for an attacker to identify the honeypots. This involves increasing the Time-to-Live (TTL) value of packets before they exit the Honeywall, the security gateway of the Honeynet. By doing this, the external interface of the Honeywall remains hidden during traceroutes from the Honeypot. Instead, the remote tunnel endpoint is revealed as the first hop, appearing as a nonlocal address. This manipulation adds a layer of complexity for attackers trying to discern the Honeynet's actual structure.

Source NAT and IP over IP Tunnels

Source Network Address Translation (NAT) can be applied to route all traffic destined for remote tunnel endpoints through the Honeypot LAN, effectively making it the default gateway. This setup ensures that the honeypots inspect all traffic. IP over IP tunnels can also prevent TTL value decrement, further masking the honeypot infrastructure. Collectively, these techniques create a more convoluted network environment, increasing the challenges for potential adversaries attempting to map the Honeynet.

In summary, distributed honeynets come in various configurations, each with its trade-offs. These architectures are designed to catch attackers in the act across multiple networks while obscuring the actual network structure. By implementing techniques such as policy-based routing, packet mangling, source NAT, and IP over IP tunnels, defenders can create intricate and challenging environments for potential adversaries. These distributed honeynet models represent a crucial step forward, especially in the context of high-interaction honeypots and advanced threat detection.

2.1.6 Risk of Honeypot and Honeynet

- **Resource Consumption:** Honeypots can consume network and computing resources. Deploying them inappropriately may lead to performance issues, affecting legitimate network traffic.
- **Deception Failures:** Skilled attackers might recognise and deliberately avoid honeypots. This could result in missed attacks or incomplete threat intelligence.
- **Legal and Ethical Concerns:** Depending on the deployment, honeypots may interact with attackers in ways that raise legal and ethical questions. For example, capturing an attacker's activity may cross boundaries regarding privacy and unauthorised system access.
- **Security Risks:** If not correctly secured, honeypots can become a point of entry for attackers into the organisation's infrastructure.
- **Criminal Activity:** If attackers can compromise the Honeynet, they can attempt criminal activity by hosting unlicensed content on the Honeynet or attacking other networks, which can be a legal issue.

- **Maintenance Costs:** Regular updates and maintenance are required to ensure that honeypots remain effective and do not pose security risks. Neglecting this can lead to vulnerabilities.
- Attackers can detect Honeynet data control and capture in the network and can potentially disable or bypass these functionalities, which can make Honeynet functionality useless and the network more prone to attacks.

2.2 Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS)

2.2.1 Intrusion Detection Systems (IDS)

An Intrusion Detection System (IDS) is a fundamental component of cybersecurity that plays a pivotal role in identifying and responding to security breaches and unauthorised access within a network or system. IDSs are designed to monitor network traffic and system activities in real-time, scrutinising them for patterns or anomalies indicative of malicious behaviour. When such behaviour is detected, IDSs generate alerts or notifications to security personnel, enabling them to take immediate action to mitigate the threat.

IDSs are categorised into two primary types:

1. **Network-Based IDS (NIDS):** NIDS systems are typically deployed at strategic points on a network, such as at the perimeter or between network segments. NIDS monitors network traffic, analysing packets and flow data to identify suspicious activity. It operates at the network level, making it capable of detecting threats that traverse the network, such as port scans, malware propagation, or unusual network patterns.
2. **Host-Based IDS (HIDS):** HIDS is deployed on individual host systems, continuously monitoring system logs, files, and activities on a specific device. It excels at detecting local threats, including unauthorised access, file integrity violations, or suspicious user behaviour.

IDS systems can also be classified into two other categories:

1. **Signature-based IDS systems** detect attacks by matching network traffic or system activity against a database of known attack signatures.
2. **Anomaly-based IDS systems** detect attacks by looking for unusual or unexpected behaviour in network traffic or system activity.

Signature-based IDS systems are very effective at detecting known attacks. However, they could be more effective at detecting new or unknown attacks. Anomaly-based IDS systems are more effective at detecting new or unknown attacks. However, they can also generate a higher number of false positives.

2.2.2 Intrusion Prevention System (IPS)

While IDSs provide valuable threat detection capabilities, they are often limited to alerting administrators or security teams after a security breach. This limitation prompted the development of Intrusion Prevention Systems (IPS), a more proactive cybersecurity solution. IPSs extend the functionality of IDSs by detecting malicious activities and taking automated actions to prevent or block them in real-time.

IPSs are designed to operate at the network perimeter and within internal network segments, strategically positioned to intercept and analyse incoming and outgoing traffic. When an IPS identifies potentially harmful

behaviour or traffic patterns, it can take predefined actions to intercept the attack. These actions may include blocking network traffic, isolating compromised systems, or modifying firewall rules to prevent further threats.

2.2.3 Evolution of IDS/IPS

Intrusion Detection and Prevention has witnessed significant evolution over the years, driven by the relentless advancements in cyber threats and technology. The evolution of IDS and IPS is divided into distinct stages:

1. **Early IDS Systems:** The inception of IDS can be traced back to the 1980s, primarily driven by the need to protect early computer networks. These early systems relied on signature-based detection, analysing known attack patterns. However, their effectiveness could have been improved, as they struggled to detect novel threats.

2. **Anomaly Detection and Machine Learning:** Modern IDS and IPS systems incorporate anomaly detection techniques and machine learning algorithms to address the limitations of signature-based detection. These systems learn the baseline behaviour of networks and systems, allowing them to detect deviations that may indicate an intrusion or threat.

3. **Integration and Automation:** Today's IDS and IPS solutions are characterised by their integration with other security tools and automation capabilities. They provide real-time threat intelligence, facilitate rapid incident response, and offer more granular control over security policies.

4. **Behavioural Analysis and Threat Hunting:** The latest evolution integrates behavioural analysis and threat-hunting capabilities. This allows organisations to proactively seek out and mitigate advanced threats that may evade traditional security measures.

In summary, the landscape of IDS and IPS has evolved significantly from basic alerting systems to advanced, proactive security solutions. Understanding this evolution and the capabilities of modern IDS and IPS is crucial for organisations striving to protect their digital assets in an increasingly hostile cyber landscape.

Chapter 3

Related Work

In the previous chapter, we discussed traditional Honeynet architectures, which typically involved the use of static honeypots, which emulate specific services and systems to entice attackers. In this chapter, we delve into advanced honeynet architectures that have evolved beyond traditional static approaches. We explore the designs, functionalities, advantages, and limitations of these architectures. Additionally, we discuss the TPOTCE honeypot tool, a notable contribution in the field of honeynet security.

3.1 Related papers

Honeynet architectures are a valuable tool for detecting and analysing cyber threats. These systems mimic real networks and lure attackers into revealing their tactics and techniques. By monitoring activity on the Honeynet, security professionals can gain valuable insights into the latest attack methods and develop effective countermeasures. However, setting up and maintaining a Honeynet can be challenging due to the complexity of the network and the need to ensure that it remains isolated from the rest of the organisation's infrastructure. This section explains different Honeynet architectures existing to detect/verify attacks in a network and reduce false positives of IDS. We discuss the implementation and drawbacks of each architecture in detail. Based on these inputs, we further discuss how Honeynet can be used to detect false positives in IDS/IPS.

3.1.1 Honeyd:

"Honeyd" [1] is an open-source, low-interaction honeypot(LIH) tool that simulates network services to deceive and track attackers. It is designed to mimic multiple computer systems on a network, creating virtual honeypots that appear as real systems with specific vulnerabilities. Honeyd can emulate various operating systems, services, and network configurations, making it a versatile deception and cybersecurity research tool. Honeyd is scalable for an extensive range of IP addresses. It can run 130,000 ports (65,535 TCP and 65,535 UDP ports), hundreds of simulated IP addresses, and entire simulated networks. It emulates only the IP stack personality of hundreds of different OSs of computers, routers and network devices. Honeyd uses a flexible configuration language that allows security professionals to customise the behaviour of virtual hosts and simulate a wide range of network environments. Configuring language can specify each virtual host's operating system, services, and vulnerabilities. Honeyd enhances its deception strategy with tar pitting abilities, effectively creating stumbling blocks that slow down and frustrate attackers, spammers, and malware. These tar pits consume the attackers' resources, deterring their activities and buying valuable time for defenders to respond. While primarily a Low Interaction Honeypot (LIH), Honeyd can facilitate high interaction by relaying traffic to an external host.

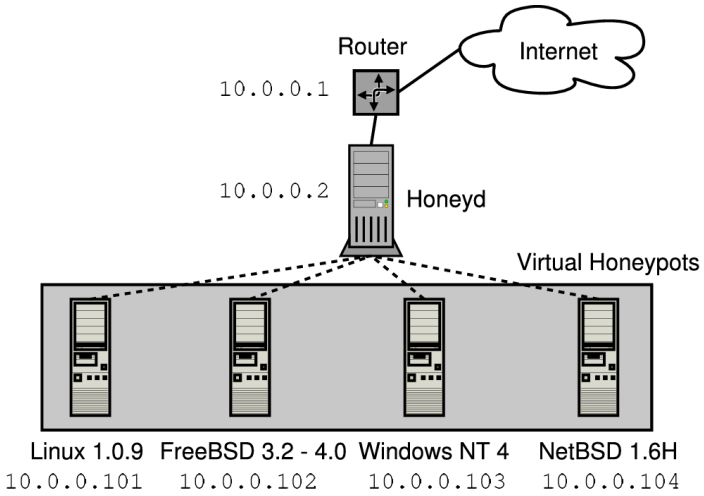

Figure : Honeyd receives traffic for its virtual honeypots via a router or Proxy ARP. For each honeypot, Honeyd can simulate the network stack behavior of a different operating system.

Figure 3.1 Honeyd Architecture

3.1.2 Niels Provos:

The paper A Virtual Honeypot Framework [28] introduces a novel approach to honeypot architecture leveraging the capabilities of Honeyd, a versatile tool for creating virtual honeypots. In this Virtual Honeypot Framework, the primary objective is to capture network traffic from a vast range of IP addresses, primarily focusing on unused IPs within the network. This traffic is then effectively diverted to the High Interaction Honeypots (HIHs), ensuring unseen first payload connections are directed towards these honeypots. Meanwhile, the Low Interaction Honeypots (LIHs) handle the remaining network traffic as Honeyd observes and manages it.

One notable feature of this architecture is its scalability. By increasing the number of Honeyd IPs, the framework enhances the probability of attackers encountering the honeypots. However, the paper acknowledges a potential limitation: if a new threat presents the same first-seen payload as previously seen traffic, the architecture may need to detect it effectively. To address the challenge, the framework employs a Control and Command Centre as a central processing unit. This command centre plays a critical role in identifying malicious traffic within the network data captured by the LIHs. It is designed to detect various threats, including worms and other potentially harmful activities.

The Control and Command Centre also manages the connection handling mechanism, including redirecting traffic to the HIHs and spawning these high-interaction honeypots. It continuously monitors the behaviour of the HIHs concerning network activity, file system behaviour, and resource utilisation. Moreover, it exercises control over the containment policy for HIHs, blocking connections after a specific number of outgoing packets, such as 5 or 10.

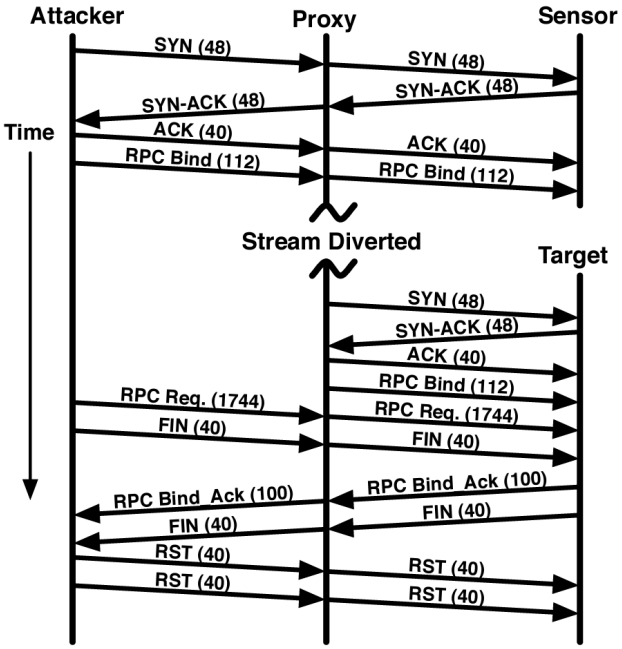

Figure 3.2 Redirection sequence diagram

One primary challenge outlined in the paper is handling new, unseen payload traffic effectively. The framework's architecture could be evaded by recent attacks that employ payloads already seen in previous traffic. To mitigate this issue, the authors employ different sampling methods on connections, enabling cross-checking by the HIHs. This approach helps identify and respond to new threats, even when they use previously seen payloads.

In summary, the Virtual Honeypot Framework presented in the paper offers an innovative solution for capturing and analysing network traffic using Honeyd and a combination of HIHs and LIHs. While it addresses scalability and detection challenges, it also recognises the need for advanced sampling techniques to enhance its effectiveness in identifying and countering evolving threats within network traffic.

Experimental numbers show that only 0.25% of packets must be sent to the backend.

3.1.3 Collapsar

The paper "Collapsar: A VM-Based Architecture for Network Attack Detention Center" [22] introduces a novel approach to network traffic redirection and attack detection. The Collapsar architecture employs router-based and end system-based redirection methods to channel network traffic to a centralised Collapsar centre for analysis and monitoring.

3.1.3.1 Router-Based Redirection

Router-based redirection uses a router supporting GRE (Generic Routing Encapsulation) tunnelling. This router redirects incoming network traffic to the Collapsar centre for further analysis. When the router receives packets destined for honeypots but lacks the MAC address of the honeypots, it queries a Redirector component located on a physical host within the same network. The Redirector responds with the MAC address, effectively

directing all packets to the Redirector. The Redirector, in turn, captures and filters the incoming traffic, encapsulates the packets, and diverts them to the Collapsar centre.

3.1.3.2 End System-Based Redirection

A Redirector component is deployed on a physical host within the production network in the system-based redirection approach. This Redirector is responsible for redirecting traffic from the production network to the Collapsar centre. When the Redirector receives traffic destined for honeypots, it captures, filters, and encapsulates the packets before forwarding them to the Collapsar centre.

3.1.3.3 Collapsar Center Functionality

The Collapsar Center is a centralised hub for analysing and monitoring network traffic. It includes a frontend component responsible for decapsulating incoming packets and forwarding them to the intended virtual honeypots. The Collapsar centre can also handle traffic from Honeypots by returning it to the Redirector.

3.1.3.4 Dynamic Creation of HIHs

High-interaction honeypots (HIHs) within the Collapsar centre are dynamically created and customised. They can log and capture live virtual images to analyse and respond to potential threats.

3.1.3.5 Gateway for Intended Traffic

The front end of the Collapsar centre acts as a gateway for facilitating intended traffic between the Collapsar centre, HIHs, and Redirectors located in various production networks. This architecture supports both server and client honeypots.

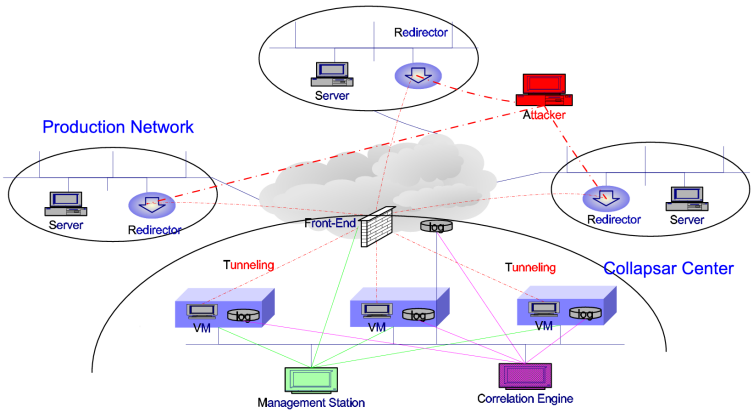

Figure : Architecture of Collapsar: a VM-based network attack detention center

Figure 3.3 Complete Architecture of Collapsar System

3.1.3.6 Overloading Concerns

A potential concern in this architecture is the overloading of HIHs by Redirectors. The paper acknowledges this issue and suggests that virtualisation technologies like VMware and UML can be used for HIH scaling. The paper utilised UML due to the unavailability of VLAN in VMware.

3.1.3.7 Delay Mitigation

The paper notes a delay may be observed in Collapsar HIH connections compared to standard production connections. This delay can be mitigated using GRE tunneling-supported routers, which can route packets to the Collapsar centre without relying on end system redirection.

3.1.3.8 Limitations

The architecture has limitations in handling encrypted traffic (e.g., SSH, HTTPS), as the Redirector cannot support filtering and encapsulation for such traffic. Therefore, encrypted traffic is sent directly to honeypots without processing. The Collapsar architecture presents an innovative approach to network attack detection and traffic redirection. It utilises both router-based and end-system-based redirection methods, offers dynamic HIH creation, and supports various network configurations. However, challenges such as potential overloading of HIHs and limitations in handling encrypted traffic should be considered when implementing this architecture.

3.1.4 Honeybrid:

The Honeybird [2] honeynet architecture represents an innovative approach to network security and threat detection. At its core, Honeybird relies on a modified Honeywall with redirection capabilities, enabling it to manage network traffic and interactions with potential attackers efficiently. Here are key aspects and features of the Honeybird honeynet architecture:

- **Traffic Decision Mechanism:** In Honeybird, the gateway plays a critical role in determining network traffic flow. This gateway is responsible for deciding which traffic should be sent to High Interaction Honeypots (HIHs) for further analysis. The Honeybird architecture enhances efficiency by utilising Low Interaction Honeypots (LIHs) to interact with potential attackers first. A decision engine within Honeybird assesses the interactions with LIHs and decides whether redirection to HIHs is necessary.
- **Redirection Criteria:** The decision to redirect traffic to HIHs is based on specific criteria, including rules associated with the target. Honeybird uses tcpdump syntax filter rules to classify the type of traffic that should be directed to the target. It employs frontend rules to determine which LIH should engage with incoming traffic, backend rules to specify which traffic from LIHs should be redirected to HIHs, and containment rules for HIHs.
- **Redirection Strategies:** The backend rule for redirection to HIHs is based on multiple factors, including randomisation, source IP, the hash of the first payload, and a counter. This complex criterion ensures that redirection is controlled and limited, preventing unnecessary overloading of HIHs. Guided by these rules, the decision engine flags connections that meet the backend criteria. The redirection engine then takes control and replicates the connection instead of immediately sending it to HIHs.
- **Containment Policy:** Honeybird implements a containment policy to restrict outgoing connections or redirect them to unaffected HIHs. This policy helps mitigate the potential impact of malicious traffic.

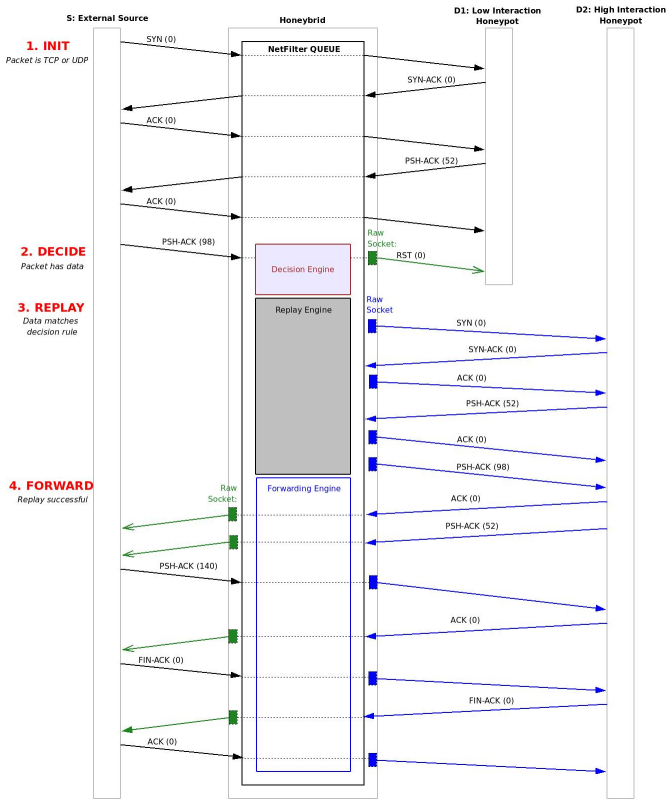

Figure 3.4 Redirection sequence diagram

- **Handling Stateful Connections:** Honeybird can manage stateless and stateful connections for redirection. It updates the checksum of UDP headers for UDP traffic, while for TCP, it maintains connection states and responds with the same sequence number to HIHs.
- **Network Scope:** Honeybird is designed to operate within a specific network segment, and its effectiveness depends on being connected to the network's ingress point. For increased coverage of all network IP addresses, changes may be required in the primary router to redirect traffic to Honeybird. This approach may not be recommended in industrial networks but can be feasible in cooperative university networks.
- **Limitations:** While Honeybird is effective in many respects, it has limitations. It does not address IP fingerprinting by potential intruders, which means that expert-level attackers may be able to detect the honeypots. Additionally, Honeybird does not handle SSH or HTTPS connections. The architecture does not include dynamic honeypot creation but can be integrated with such functionality.
- **Delay Considerations:** The paper discusses the delay introduced by the Honeybird architecture, providing insights into the potential impact on network performance.

In summary, the Honeybird honeynet architecture is a sophisticated system employing a gateway-based approach to manage and redirect network traffic for security analysis effectively. While it offers notable benefits, such as scalability and efficient traffic redirection, it also has limitations related to IP fingerprinting and specific types of network connections. Understanding these nuances is essential when deploying Honeybird in a network security environment.

3.1.5 Ip fingerprinting:

The paper titled ”Dynamic Hybrid Honeypot System Based on Transparent Traffic Redirection Mechanism” [4] introduces a novel approach to honeynet architecture that addresses the issues of latency and IP fingerprinting inherent in previous systems like Honeybird. This innovative model leverages dynamic honeypot generation and transparent traffic redirection to enhance its effectiveness in detecting and deterring potential attackers.

3.1.5.1 Latency and Detectability

The paper acknowledges that Honeybird, while effective, has reasonable latency that skilled attackers can detect. Furthermore, attackers can identify honeypots by checking the IP address of the final compromised system, revealing the presence of honeypots. These limitations necessitate the development of a more robust and stealthy honeynet architecture.

3.1.5.2 Dynamic Honeypot Generation

This approach involves the creation of honeypots on the fly, as needed, to respond to potential threats. The Redirection engine is crucial in managing this dynamic honeypot generation process.

3.1.5.3 KVM-Based Spin-Up

The paper describes using KVM (Kernel-based Virtual Machine) to spin up suspended High Interaction Honeypots (HIHs). When a connection is flagged for redirection, the Redirection engine initiates the creation of the corresponding HIHs. The spin-up process takes approximately 1.5 seconds, ensuring a rapid response to potential threats.

3.1.5.4 Honeyd for LIHs Generation

Low Interaction Honeypots (LIHs) are generated using Honeyd, a versatile tool for creating virtual honeypots. Honeyd allows for emulating various services and systems, enhancing the deception factor and attracting a wide range of potential attackers.

3.1.5.5 VMX for HIHs and Virtual Networks

HIHs, as well as the creation of virtual networks, are managed using VMX. This technology provides the necessary infrastructure for rapidly deploying and managing HIHs. Additionally, VMX enables the seamless switching of virtual honeypots from one state to another, facilitating dynamic honeypot generation.

3.1.5.6 VNX for Scenario Reconfiguration

VNX, an integral part of the proposed model, can reconfigure honeypot scenarios using XML syntax-based reconfiguration files. This flexibility allows for the dynamic adjustment of honeypot behaviour without disrupting the overall network environment.

3.1.5.7 LXC-Based VNX

The paper discusses using LXC (Linux Containers)-based VNX, which is particularly well-suited for emulating Linux operating systems. While it excels in minimal startup delay (less than 1 second), it may lack fidelity when simulating non-Linux operating systems.

3.1.5.8 Customizable Standards

The proposed model offers customisable standards within VNX boxes, allowing adjustments based on the desired efficiency levels. However, it is essential to note that increased customisation may come with higher investment costs, which could concern smaller institutions.

3.1.5.9 Conclusion

In conclusion, the "Dynamic Hybrid Honeypot System Based on Transparent Traffic Redirection Mechanism" paper introduces an advanced honeynet architecture that overcomes the challenges of latency and IP fingerprinting. By employing dynamic honeypot generation, transparent traffic redirection, and a combination of virtualisation technologies like KVM, Honeyd, VMX, and VNX, this system offers an efficient and adaptable solution for detecting and countering potential cyber threats while minimising the risk of detection by attackers. However, it is essential to consider the trade-offs, such as customisation costs, when implementing this architecture in different network environments.

3.1.6 Potemkin Virtual Honeyfarm:

The paper "Scalability, Fidelity, and Containment in the Potemkin Virtual Honeyfarm" [12] discusses the problem of scalability and containment in the context of honeypots. Conventional honeypot networks could be more efficient, wasting processor cycles and memory resources due to low utilisation. Additionally, maintaining a joint state and executing common code paths across multiple honeypot servers leads to duplication of effort. The paper aims to address these inefficiencies and improve the scalability and containment of honeypot networks.

3.1.6.1 Architecture

The Potemkin Virtual Honeyfarm is designed to be a centralised honeynet comprising a collection of monitored internet hosts, each running a standard operating system and application software. The key innovation lies in using GRE Tunneling configured on routers, allowing external incoming packets to be tunnelled to the Honeyfarm gateway. At this gateway, packets are filtered and modified before being sent to the Honeyfarm.

The Honeyfarm gateway employs filters to prevent unnecessary virtual machines (VMs) from spawning due to scans, worms, viruses, and other disruptive activities. These filters can be based on known signatures of worms, viruses, or proxies. If a more substantial transaction is detected, the traffic is migrated to a dedicated VM for further analysis.

Notably, the virtual honeypots within the Honeyfarm do not possess an IP address but have a MAC address. The gateway maintains detailed records of active connections, past mappings, and other vital information to preserve the illusion that a particular IP address hosts a specific software configuration.

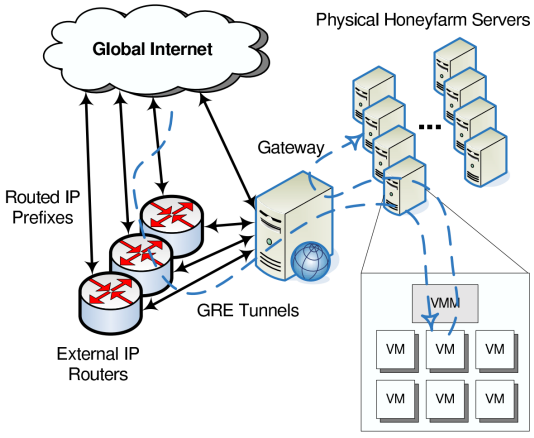

Figure 3.5 Honeyfarm architecture for Potemkin virtual Honeyfarm

3.1.6.2 Packet Handling and Containment:

The Potemkin architecture includes mechanisms for efficient packet handling and containment. When a packet arrives, the system checks the cache table to determine if the corresponding VM is already allocated. If not found in the cache table, it is looked up in the history table, which maintains long-term state information about source/destination and timestamps. This information calculates the load and decides the appropriate course of action.

In cases where a packet is not found in either the cache or history tables, the gateway redirects the traffic to a less-loaded Honeyfarm of the same host type by changing the MAC address in the packet. The worst-case delay caused by this redirection is capped at 40 seconds. ‘

3.1.6.3 Universe Concept for Containment

To prevent cross-contamination between distinct contagions within the same HIH VM (Honeyfarm Individual Honeypot), the concept of ”universes” is introduced. Each internet packet is initially assigned to ”Universe-0.” New universe IDs are created within the Honeyfarm for each combination of (source, destination, and source port).

Internal reflections keep track of the universe ID via the history table. This ensures that an outbound packet is subject to the containment policy of the initial targeted IP address. This approach ensures that containment policies for each IP address are specific and redirected to the corresponding universe ID HIH, maintaining strict containment.

3.1.6.4 Scalability Optimization

The Potemkin paper also focuses on optimising the Honeyfarm for scalability. It achieves this by leveraging the Copyonwrite XEN virtualisation software’s (COW) and shadow cloning features. COW reduces memory requirements by sharing memory pages of the base OS among virtual machines. The Virtual Machine Monitor (VMM) handles the spawning and management of connections until the VM is ready.

However, achieving scalability required modifications to the XEN source code and adjustments to routers using CLICK. The system can support up to 1,500 virtual machines on a single physical device, but it is limited to paravirtualisation-supported OS and does not support Windows guest systems.

The scalability of a Honeyfarm server depends on three key overheads:

1. Memory overhead required by honeypot VMs.
2. CPU overhead is required to create honeypot VMs.
3. CPU utilisation of honeypot VMs responding to traffic.

The authors introduce an inactive timeout for honeypots to control CPU utilisation. The VMM creates a new VM for each new IP address that arrives. A 60-second idle timeout turns off active honeypots, preventing Honeyfarm overload.

The paper highlights that the delay caused by flash cloning of the VM, running the VM, and generating the response is initially 521 ms. However, this delay can be significantly reduced to 325 ms by modifying the data structure with prealloc, avoiding the need to tear down the VM completely.

3.1.6.5 Conclusion

The Potemkin Virtual Honeyfarm is a groundbreaking approach to honeypot deployment, offering an effective solution for capturing cyberattacks with high fidelity while ensuring scalability and containment. By combining virtualisation techniques, careful packet handling, and the introduction of the universe concept, the Potemkin architecture presents a compelling advancement in the field of cybersecurity. Furthermore, the gateway's capability to handle 28,000 packets per second with random traffic underscores its practical applicability and efficiency in real-world scenarios. Potemkin Virtual Honeyfarm does not deal with what kind of traffic should be directed to Honeynet. It only deals with Honeynet management and how High interaction Honeypot can be spawned efficiently.

3.1.7 Anomaloy detectors:

The paper "Detecting Targeted Attacks Using Shadow Honeypots" [13] presents an innovative approach to cybersecurity by combining honeypots with anomaly detection techniques to capture client-side attacks, which can be detected algorithmically.

3.1.7.1 Honeypot Coupling

In this architecture, honeypots are tightly coupled or loosely coupled to a server or client. Tightly coupled honeypots mirror functionality and state in a way that makes it indistinguishable from the legitimate system, making it challenging for attackers to differentiate. On the other hand, loosely coupled honeypots do not mirror the state. They are primarily used to detect generic attacks, such as static attacks that do not rely on user behaviour triggers.

3.1.7.2 Anomaly Detection Techniques

Abstract Payload Detection (APD) and payload shifting are the anomaly detection techniques used for experimental results in this paper. Traffic filtering is the first step, accomplished by filtering components that consider

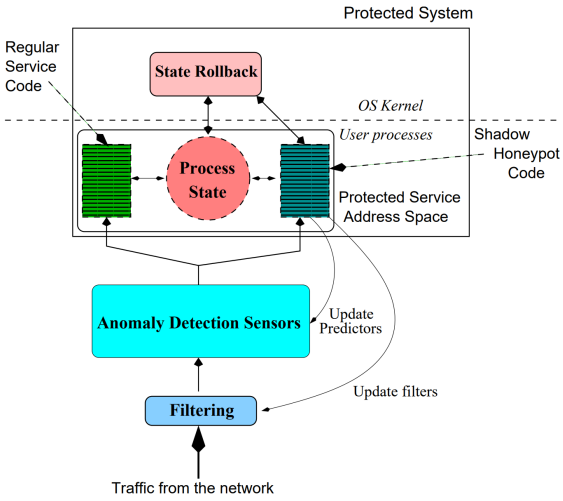

Figure 3.6 Shadow Honeypot architecture.

payload content, source IP, and other attributes to block known attacks. The filtered traffic is then passed to anomaly detectors, finely tuned to increase detection accuracy. Any traffic identified as potentially dangerous by the anomaly detection system is subsequently sent to honeypots for verification. Traffic that passes through the anomaly detectors without raising alarms is directed to a typical server instance.

3.1.7.3 Detection of Server and Client Attacks

For server attacks, anomaly detectors are placed in an HTTP proxy that employs APD to redirect traffic. However, for client attacks, especially those targeting passive attacks that do not rely on user behaviour triggers, using a proxy is not feasible due to added latency. Instead, sensors decode the HTTP protocol in TCP stream reconstruction to extract suspicious objects. These sensors can select a single packet or an entire flow, or sets of flows, to redirect to honeypots. These detectors are equipped with advanced methods to detect attacks and roll back to the initial state, as they are less frequently used.

3.1.7.4 Performance of Shadow Honeypots

The shadow version of the Apache server included in this architecture can handle up to 1300 requests per second, while the Mozilla Firefox client shadow version can take 1 to 4 requests per second. Anomaly detectors are carefully tuned to ensure they do not produce excessive alerts and consider the required false positive rate. Shadow honeypots rely on anomaly sensors to explicitly divert traffic related to the type of attacks they aim to detect.

3.1.7.5 Drawbacks and Considerations

This method, while promising, has its drawbacks. The latency of detectors in shadow honeypots is associated with the cost of implementing state rollback to the initial state. Rollback costs are incurred when an attack is

successfully detected or the system state remains unchanged. The bottleneck in this architecture is the anomaly detectors, which should be appropriately configured to prevent bypassing.

In conclusion, "Detecting Targeted Attacks Using Shadow Honeypots" introduces a sophisticated approach to cybersecurity by combining honeypots and anomaly detection techniques. This approach aims to capture and mitigate targeted client-side attacks algorithmically, improving network security. However, carefully configuring and considering latency and resource costs are essential for successfully implementing this architecture.

3.1.8 Shark: Spy Honeypot with Advanced Redirection Kit

3.1.8.1 Introduction:

Shark [16] is an innovative honeypot framework that offers advanced redirection capabilities for enhanced network security. It leverages a dynamic redirection mechanism that effectively deceives attackers by directing suspicious traffic to designated honeypots for further analysis. Unlike traditional honeypots that rely on static redirection, Shark's dynamic approach ensures that malicious connections are always redirected to the appropriate honeypot, enabling comprehensive monitoring and analysis of attack behaviour.

3.1.8.2 Key Features:

Shark's dynamic redirection mechanism is its core strength, providing several advantages over traditional honeypots:

- Attack Diversion and Containment: Shark effectively diverts suspicious traffic from the production network, preventing potential attacks from reaching critical systems.
- Honeypot Resource Optimization: Shark directs traffic to the most appropriate honeypot based on the attack type, ensuring that honeypot resources are utilized efficiently.
- Attack Behavior Analysis: Shark's redirection mechanism enables comprehensive analysis of attacker behaviour, providing valuable insights into attack patterns and tactics.

3.1.8.3 Implementation:

Shark's redirection mechanism is implemented using a combination of techniques:

- Packet Routing: Shark modifies routing tables to redirect suspicious traffic to specific honeypots.
- IP Address Manipulation: Shark modifies IP addresses within packets to direct traffic towards designated honeypots.
- Port Rewriting: Shark rewrites TCP/UDP port numbers to ensure that connections are established with the appropriate honeypot services.

3.1.8.4 Dynamic Redirection :

Shark's dynamic redirection mechanism is based on a real-time network traffic analysis. It employs a set of rules to identify suspicious traffic and determine the most appropriate honeypot for redirection.

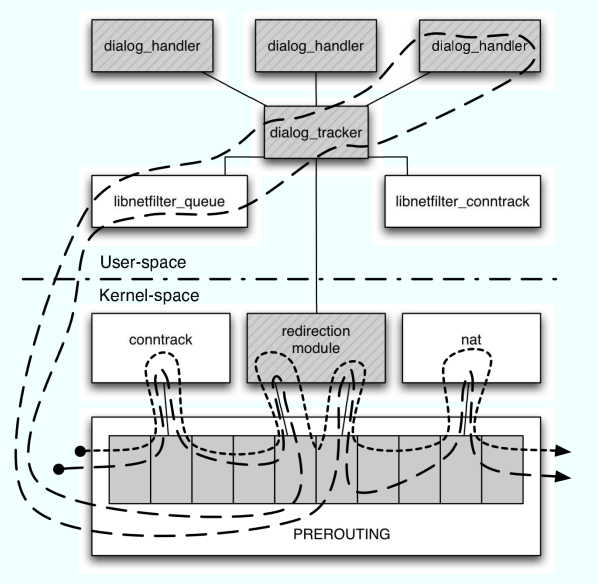

Figure 3.7 Mechanism of redirection

Shark utilizes netfilter, a firewall component in the Linux kernel, to dynamically redirect traffic between honeypots. It employs two hooks within the PREROUTING chain of Netfilter to achieve this redirection:

- Extraction Hook: This hook intercepts and extracts the first packet of a connection. It then sends the extracted packet to the dialog_tracker, a user-space module responsible for analyzing the packet and determining whether redirection is necessary.
- Tagging Hook: If the dialog_tracker decides to redirect the connection, it informs the kernel module via a netlink socket. The tagging hook then marks the corresponding connection as ”redirected”.

Once a connection is tagged as ”redirected”, the nat dst hook in the PREROUTING chain modifies the packet's destination address to redirect it to the designated honeypot. This redirection is configured using the iptables command.

These rules for redirection are based on various factors, including:

- Attack Type: The shark identifies the type of attack based on pattern matching and heuristic analysis.
- Attack Source: Shark identifies the attacker's source IP address and geolocation.
- Target Protocol: Shark identifies the protocol being used by the attacker, such as HTTP, FTP, or SSH.

Based on this information, Shark dynamically redirects traffic to the most suitable honeypot for further analysis.

3.1.8.5 Benefits:

Shark's advanced redirection capabilities offer several benefits for network security: Reduced False Positives: Shark's dynamic redirection mechanism reduces the number of false positives by directing only truly suspicious traffic to honeypots. Enhanced Attack Detection: Shark effectively detects a wide range of attacks by redirecting traffic to honeypots that are specifically designed to handle them. Improved Attack Analysis: Shark provides an

in-depth analysis of attack behaviour, enabling security teams to gain valuable insights into attacker tactics and motivations. Optimized Honeypot Utilization: Shark's dynamic redirection ensures that honeypot resources are utilized efficiently by directing traffic to the most appropriate honeypot for each attack type.

3.1.8.6 Conclusion;

Shark's dynamic redirection mechanism differentiates it from traditional honeypots, offering enhanced network security and attack detection capabilities. Its ability to effectively divert suspicious traffic, analyze attack behaviour, and optimize honeypot utilization makes it a valuable tool for security teams seeking to protect their networks from a wide range of cyberattacks.

3.2 Potential Risks and Challenges of Honeypots and IDS/IPS Systems

3.2.1 Risks:

- **Network Complexity:** Managing honeypots within an IDS/IPS system adds complexity to the network. Configuration, monitoring, and maintenance of honeypots require expertise and effort.

- **False Positives:** Honeypots are designed to attract attackers, which can generate false alarms. An IDS/IPS system that relies heavily on honeypots may produce more false positives, which can cause more noise and resource wastage.

3.2.2 Challenges:

- **Scalability:** As network size and complexity grow, ensuring that honeypots effectively cover all potential attack vectors can be challenging. Scaling honeypot architectures to cover the entire network requires careful planning.

- **False Positives:** The challenge of reducing false positives introduced by honeypots lies in effectively configuring and fine-tuning IDS/IPS systems and Honeypots. When discussing False positives, False negatives come into the picture. Verifying Honeypot's False negative rate is a challenging part.

- **Resource Management:** Effectively managing the resources required for deploying and maintaining honeypots is critical. Resource allocation, especially in large architectures, is a significant concern.

- **Mitigating Risk:** Addressing the risk of honeypot compromise requires advanced security practices. This involves monitoring honeypots for signs of intrusion, promptly updating and patching them, and segregating honeypots from critical production networks.

- **Latency:** Depending on the architecture, honeypots may introduce latency into network traffic, impacting the user experience and application performance. Minimising this latency is essential in specific environments. For example, many architectures mentioned above utilise a redirection mechanism, which is a potential increase in latency which attackers can detect.

3.3 Honeycomb:

In this section, we discuss a tool which uses Honeypot data to enhance the NIDS signatures. Honeycomb [17] is an innovative system that combines protocol analysis and pattern-detection techniques to automatically generate signatures for network intrusion detection systems (NIDSs). It focuses on malicious traffic, specifically worms, and can generate detailed signatures for well-known worms like Slammer, Code Red and THCIISSLAME.c. Honeycomb also adapts to various traffic types and can generate signatures for new or lesser-known programs.

One of the critical strengths of Honeycomb is its ability to integrate with the Honeyd honeypot system. This integration eliminates cold-start issues and allows Honeycomb to analyze and generate signatures actively, enhancing network threat detection and response capabilities. Here are some examples of how Honeycomb can be used:

1. **Potential applications:** Honeycomb can be applied to any traffic to search for signatures when unavailable. Spam detection is one possible use case.

2. **To detect new or unknown worms:** Honeycomb can generate signatures for new or lesser-known worms, which can help organizations protect themselves from these threats before they become widespread.

3. **To improve the accuracy of NIDS systems:** Honeycomb can generate more detailed and accurate signatures than traditional methods, which can help reduce the number of false positives and negatives generated by NIDS systems.

4. **To monitor for targeted attacks:** Honeycomb can generate signatures for specific attack patterns, which can help organizations detect targeted attacks targeting their specific systems and networks.

3.4 T-POT CE (Telekom Security's Honeypot Platform Community Edition)

T-POT CE (Telekom Security's Honeypot Platform Community Edition) [6] is an innovative solution designed to streamline the deployment and management of honeypots, offering a cost-effective and resource-efficient approach. Deploying and managing honeypots, including Medium Interaction Honeypots (MIHs) and High Interaction Honeypots (HIHs), can be resource-intensive and often dependent on specific operating systems. T-POT CE leverages Docker technology to simplify this process by allowing applications to share the same Linux kernel as the host system, eliminating the need to create separate virtual operating systems.

3.4.1 Architecture

T-POT CE, which is Debian-based and dockerised, is a versatile platform that runs multiple honeypot daemons and associated software tools within Docker containers. Each honeypot operates within its isolated environment, ensuring security and containment. By default, T-POT CE comes pre-equipped with a wide range of honeypots and tools. Furthermore, the platform allows new honeypots to be added seamlessly using Docker Compose files.

3.4.2 Honeypot Categories

One of the critical strengths of T-POT CE lies in its extensive selection of honeypots, categorised into three types based on the level of interaction they offer to potential attackers. Provides a comprehensive range of services, which are divided into five main groups:

1. **System Services:** These services are provided by the operating system and include essential functionalities.
    - **SSH:** Secure Shell for secure remote access.
    - **Cockpit:** A web-based remote access, management, and web terminal interface.
2. **Elastic Stack:** T-Pot CE incorporates the Elastic Stack, a powerful suite of tools for managing and analysing data.
    - **Elasticsearch:** Used for storing events and data.
    - **Logstash:** Responsible for ingesting, receiving, and sending events to Elasticsearch.
    - **Kibana:** Provides beautifully rendered dashboards for displaying events and data.
3. **Tools:** T-Pot CE includes various useful tools to enhance its capabilities.
    - **NGINX:** Offers secure remote access via reverse proxy to Kibana, CyberChef, Elasticvue, GeoIP AttackMap, and Spiderfoot.
    - **CyberChef:** A web application for encryption, encoding, compression, and data analysis.
    - **Elasticvue:** A web frontend for browsing and interacting with an Elasticsearch cluster.
    - **T-Pot Attack Map:** A beautifully animated attack map specific to T-Pot.
    - **Spiderfoot:** An open-source intelligence automation tool.
4. **Honeypots:** T-Pot CE includes a selection of 22 available honeypots, which can be customised based on the chosen edition and setup.
5. **Low-Interaction Honeypots:** Examples include MedPot, Maloney, Honeyd, Heralding, Endlessh, ElasticPot, Dionaea, DDOSpot, Conpot, CitrixHoneypot, Cisco ASA, ADB Honey and IPP Honey. These honeypots simulate minimal interaction to lure and capture attackers without exposing significant vulnerabilities.
6. **Medium-Interaction Honeypots:** This category encompasses honeypots like SentryPeer, DICOMPot, HoneyTrap, Glutton Snare, and Tanner. They provide moderate interaction with attackers, allowing for a more realistic simulation of services and systems.
7. **High-Interaction Honeypots:** High-interaction honeypots, such as Cowrie, Hellpot and Redishoneypot, offer the closest emulation to real systems and services. They allow in-depth interaction with attackers, capturing more sophisticated attack techniques.
8. **Network Security Monitoring (NSM):** T-Pot CE incorporates NSM tools for enhanced network security monitoring.
    - **Fatt:** A pyshark-based script for extracting network metadata and fingerprints from pcap files and live network traffic.
    - **P0f:** A purely passive traffic fingerprinting tool.
    - **Suricata:** A Network Security Monitoring engine.

3.4.3 Resilience and Responsiveness

By offering this diverse range of honeypot interfaces, T-POT CE effectively collects and manages data within a Honeynet while combining the advantages of both low and high-interaction honeypots. Additionally, T-POT CE can rapidly restart or create new honeypot instances with minimal resource consumption, which is invaluable when responding to severe attacks. This feature ensures the honeypot environment remains resilient and responsive even under high-stress situations.

3.4.4 Conclusion

In summary, T-POT CE is a powerful and comprehensive honeypot platform that leverages Docker technology to simplify the deployment and management of honeypots. It offers a wide selection of honeypots with varying levels of interaction to effectively capture and analyse malicious activities while optimising resource usage and providing flexibility in the face of adversarial threats.

Chapter 4

Design Architecture

In the preceding section, we observed that traditional Honeynet architectures are often complex regarding deployment, maintenance, and resource requirements. Typically, these architectures involve deploying multiple physical honeypots, which can be cumbersome and costly. While some architectures incorporate virtual machine-based honeypots, they often come with proprietary configurations, adding to the overall complexity. This complexity has hindered organisations' effective utilisation of honeypots as a security measure. This chapter introduces a deployable architecture using open-source tools, avoiding intricate modifications.

The core idea is to verify IDS/IPS alerts for false positives using a Honeynet, utilising the architecture of Shadow Honeypot discussed in the previous chapter. The "Detecting Targeted Attacks Using Shadow Honeypots" architecture leverages anomaly detectors to identify attacks and redirects them to shadow honeypots. While effective, this system has drawbacks, including complexity in creating attack-dependent shadow honeypots and potentially expensive state change rollback mechanisms. To overcome these drawbacks, the current architecture incorporates Honeybrid. Honeybrid's efficient traffic redirection seamlessly reroutes suspicious traffic flagged by IDS/IPS to the Honeynet, offering a more scalable and cost-effective solution.

4.1 Architecture:

4.1.1 Components:

The current architecture comprises three primary functional components, as shown in 4.1: The IDS/IPS, the Honeybrid, and the Honeynet. These components work together to provide better automation and run-time detection of false positives.

4.1.1.1 IDS/IPS:

It is co-located with the Honeybrid system. IDS/IPS has two main functions: detecting malicious traffic based on signatures or rules and sending logged alerts to the Honeybrid. The role of IDS/IPS is pretty simple when compared to the other two functional components. The current implementation is more suitable for anomaly-based IDS/IPS, as false positives are more frequent in those systems.

4.1.1.2 Honeybrid:

A centralised Honeynet architecture that bridges the production network and Honeypots. Decide to redirect traffic to Honeynet transparently based on alerts from IDS/IPS. If there is no alert, traffic is, by default, sent to

the server through Honeybrid. The Honeybrid gateway is the central part, including the Decision Engine and the Redirection Engine, which orchestrate the filtering and redirection to the Honeynet. The Decision Engine is used to select interesting traffic based on the IDS/IPS alerts, and the Redirection Engine is used to transparently redirect the traffic from the actual network to the Honeynet farm.

4.1.1.3 Honeynet:

Honeynet can consist of High Interaction Honeypots (HIHs), Low Interaction Honeypots (LIHs), Medium Interaction Honeypots(MIHs) and tools that help in detecting attacks together. The Honeynet task here is to verify the malicious activity detected by IDS/IPS. As it is a centralised Honeynet architecture, Honeynet can be outsourced to a cloud in a different network, which is an advantage of the current architecture. This enhances the security and management of the overall system. A virtual Honeynet is recommended for this architecture to address scalability and resource overhead. TPOT CE is used to implement Honeynet in our experiment.

Incoming traffic to the production server passes through the gateway, where IDS/IPS passively or actively listens to the ingress interface and matches the packet with the signature. If the packet matches, an alert is logged and sent to Honeybrid. Honeybrid, on the other hand, receives the IDS/IPS alert and decides to redirect traffic to Honeypot in Honeynet based on the alert. If there is no alert, traffic is sent to the server without any default redirection.

When the Honeybrid redirects traffic to the Honeynet, it modifies the routing table on the gateway. This routing modification forces the gateway to send all traffic directed at the server to the Honeynet instead. We will discuss routing changes in more detail in the implementation subsection 5.1.5. The Honeynet then interacts with the attacker and collects information about the attack. This information is used to identify false positives in the detection of IDS/IPS.

The architecture supports server-side and client-side attacks by redirecting all alerted traffic to the Honeynet. We have seen how server-side attacks are redirected to the Honeynet. Similarly, if an attacker is trying to exploit a vulnerability in the client, let's say, a browser like an HTTP client present inside the network, which triggers the IDS/IPS system when a malicious website is accessed, will generate an alert. Honeybrid will then redirect all traffic directed at the client to the Honeynet.

4.2 Data Control:

The important aspect of any Honeynet architecture is Data Control +

Data control can be done at Honeybrid and Honeynet using this architecture. We will look at more implementation details in the next chapter 5.1.5.4.

- The Honeybrid includes a mechanism for rate-limiting network packets to safeguard the actual server from potential attacks due to redirection delays. This rate-limiting control restricts traffic flow and protects the production network from adverse effects related to redirection processes.

- A control rule is applied to limit the number of connections initiated by Honeypots at Honeybrid as a further measure to control data and protect the outside network from Honeynet. If an attacker manages to compromise a Honeypot, the control rule acts as a safeguard to prevent excessive connections and mitigate potential risks.

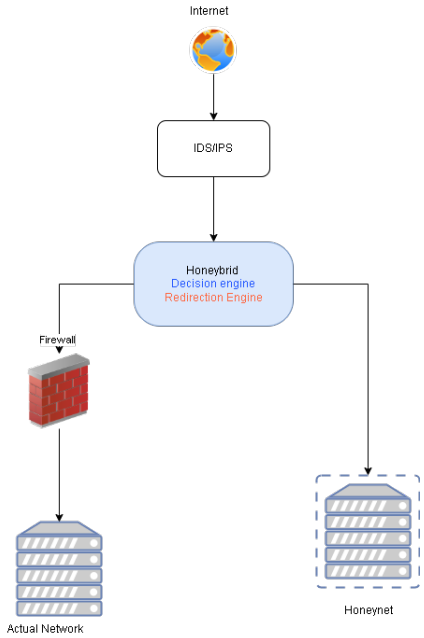

Figure 4.1 Proposed Architecture

4.3 Data Capture:

In the current architecture, we have three potential data capture points.

- IDS/IPS alerts capture network traffic records and keep track of the communication metadata like to and from IP addresses, ports, protocol, etc, along with the content matching the malicious signature or behaviour.
- The Honeybrid transaction record keeps track of the redirected traffic metadata like IP, ports, protocol, timestamp, etc, along with redirection decisions. It also logs data when Honeynet attempts to connect outside the network.
- Finally, we have logs from Honeynet, which capture detailed levels of the attacks and the malicious software/scripts used to perform the attacks. They provide much more valuable information about attacks than all other data capture combined.

4.4 Advantages Over Other Architectures:

Compared with related architectures mentioned in the previous chapter, the current architecture uses Docker, a lightweight virtualised solution. This reduces the Honeypot spinup time and resources very much.

4.4.1 Over Collapsar:

- Tpot avoids potential overloading issues of HIHs (High Interaction Honeypots) that Collapsar may experience. Collapsar supports only HIHs. The ”mod hash” module accepts packets with a new original payload,

computes a hash value for every payload inspected, and keeps a database of known payloads. When the number of HIH instances exceeds the tree height limit, Honeybrid initiates mitigation techniques (redirecting to LIHs instead of HIHs) to reduce the overall load in TPOTCE.

- Collapsar incurs additional overhead for redirection. Furthermore, it causes undesirable cross-talk between honeypots, which logically belong to different production networks. Synthetic cross-talk may decrease the authenticity of Collapsar architecture.

4.4.2 Over Anomaly-Based Honeypots:

- HTTP traffic was tested with Shadow Honeypots using an HTTP proxy to redirect. However, the redirection mechanism should have been discussed for other traffic profiles such as UDP, SSH, FTP, etc. The current architecture discussed has a more versatile and comprehensive approach to different types of network traffic using Honeybrid, making it suitable for a broader range of scenarios compared to Shadow Honeypots, which were explicitly tested only with HTTP traffic.
- Shadow Honeypots are hard to implement as they use hybrid functional and rule-based language to convert source code to Honeypot.
- If the attack is detected, the rollback mechanism of the shadow honeypot is also costly.

4.4.3 Over Potemkin:

- Potemkin Honeyfarm architecture requires modifications to the XEN source code and adjustments to routers using CLICK.
- Potemkin Honeyfarm architecture is optimised only for HIHs by XEN software modification, copy-on-write and shadow cloning to reduce the Honeypot spinup time and resource utilisation. On the other hand, this optimisation is applicable only for XEN-based virtualisations.

4.4.4 Over Shark:

- Tpot offers a dynamic honeypot generation capability in the Honeynet, while Shark paper deals only with a single Nepenthes Honeypot and does not talk about the scalability of Honeynet architecture.
- Dynamic redirection in SHARK requires kernel modifications to add hooks to Netfilter, whereas Honeybrid handles this flawlessly using rules.
- SHARK architecture was designed to detect botnets and has not been tested for other sophisticated attacks.

4.4.5 Over Dynamic Hybrid Honeypot System Based on Transparent Traffic Redirection Mechanism:

- The Dynamic Hybrid Honeypot System uses KVM-based Honeypot generation, which, on average, takes 1.5 seconds for a spinup suspended Honeypot and high resource utilisation. The current architecture uses dynamic honeypot generation using docker-based containers, which takes less than 1 second for spinup with minimal resources. This further can be reduced by having prebuilt containers readily available when Honeypot generation is required.

- In the Dynamic Hybrid Honeypot System, the Honeybrid must switch off the LIH and spin up the corresponding HIH to redirect traffic to HIH. Current architecture redirection is done from the actual network to Honeypot smoothly using Honeybrid capabilities of VLAN without the need for Honeybrid to handle it separately.
- Dynamic Hybrid Honeypot System does not support HTTPS and SSH protocols.

4.5 Advantages:

- **Real-time detection of false positives:** Honeypots can verify the malicious activity detected by IDS/IPS in real-time.
- **Better utilisation of Honeypot resources:** Honeypots can be used to collect more data about the alert, which can be used to improve the accuracy of IDS/IPS rules.
- **Reduced false positives:** Over time, attack signatures or rules can be tuned to have fewer false positives with the help of Honeypot detection.
- This architecture leverages readily available open-source tools like Honeybrid and Honeynet(TPOT CE), eliminating the need for complex custom configurations and reducing deployment and maintenance costs.
- Current architecture can be easily scaled to accommodate growing network traffic or diverse attack vectors by adding or removing Honeypots in TPOT CE and Honeybrid rules. This scalability makes it adaptable to evolving security needs.
- Honeynet components can be outsourced to a cloud environment, offering improved security, scalability, and management flexibility.

4.6 Limitations:

The current architecture has some limitations. If an alert is a false positive, the user may experience choking as they receive a response from the Honeypot. While this behaviour is acceptable during an incident, Honeycomb described in section

Furthermore, privacy laws prohibit the real-time photography of people's communications without their knowledge, as it violates their privacy. Security technologies used to protect or secure the environment are exempt from these privacy restrictions, such as the IDS sensor.

Honeypots, like IDS, firewalls, and network sniffers, are tools that can have numerous vulnerabilities. Attackers can exploit these vulnerabilities to attack other systems using honeypots, which can lead to civil liability. To mitigate this liability, outgoing traffic from the Honeypot can be redirected to another instance instead of the intended user.

Chapter 5

Implementation

This chapter delves into the technical details of implementing each component explained in the Architecture. We will also look into different design aspects considered in implementing each component.

Implementing the architecture described in the previous chapter involves using Snort as the Network Intrusion Detection System (NIDS), Honeybrid for transparent traffic redirection and containment and TPOT CE as Honeynet for detecting attacks. The implementation utilizes Virtualbox to create virtual machines to simulate the network environment.

The Honeynet should be placed in separate VLANs from the Apache server to secure the network from malicious traffic. However, for the sake of implementation, we implemented all components in the same VLAN network. Below is a detailed explanation of the implementation:

5.1 Components:

5.1.1 Apache server:

The Apache server is used as the Production server where actual traffic is intended. We hosted a simple University templated website on the Apache web server. This website contains different web pages along with a login page. Virtualbox2 network interface is assigned an IP 10.0.0.2.

Below is the table 5.1 is the configuration of the Apache web server used.

Table 5.1 Configuration of Apache Server

Apache VirtualBox Configuration	
Operating System	Ubuntu 22.04.3 LTS
RAM	1 GB
Processors	4
Storage	10GB
Network interface	NAT

5.1.2 Gateway:

Honeybrid and Snort are located on Gateway, which will be listening to the NAT interface for the incoming traffic. The Apache web server and TPOT CE reside in the VirtualBox1 NAT interface as per figure 5.1. As per

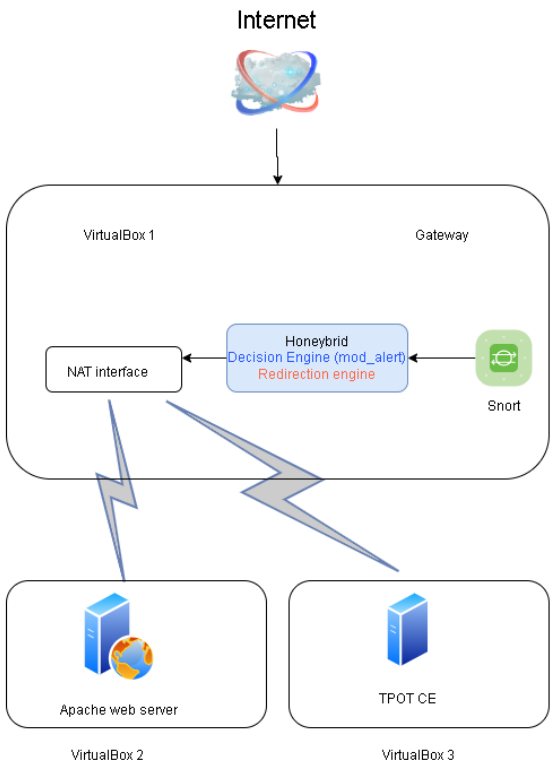

Figure 5.1 Architecture design of Implementation

our experiment, traffic intended for the Apache web server is received on the NAT interface of virtualbox1 as it is in the NAT network. In case the Honeybrid doesn't reside in the gateway, we can have an ingress router of the network port forward the traffic to Honeybrid, which is on the same NAT network as the Apache server and TPOT CE.

5.1.3 Snort (NIDS):

Snort is a popular choice for NIDS because it is free, open-source, and feature-rich. It can detect various attacks, including denial-of-service attacks, malware infections, and unauthorized access attempts. Snort is also highly customizable, allowing organizations to tune the rules to meet their needs. Snort is deployed as the network intrusion detection system and is responsible for passively listening to incoming traffic on the network.

The latest Snort version, 2.9.2, is installed, and then the conf file is configured with the external and home interface IP addresses. We configured snort with the latest version of the community snort ruleset. Below in table 5.2 is the configuration of the VirtualBox container on which Snort runs.

Table 5.2 Configuration of Snort

Snort VirtualBox Configuration	
Operating System	Ubuntu 22.04.3 LTS
RAM	8 GB
Processors	4
Storage	100GB
Network interface 1	Bridged adapter
Network interface 2	NAT

5.1.3.1 Alerting mechanism:

Snort fast mode is used as we only need metadata for redirection in Honeybrid. Snort alerts report the timestamp, send an alert message, and show the source IP address, port, destination IP address, and port. This mode is instructed using the -A fast flag. As for the ease of implementation, Snort and Honeybrid are running on the same system, which makes Honeybrid listen to snort alerts faster in real time. We used the Snort Unsock alert method to export alert reports to other programs through Unix sockets. The unsock mode is implemented by adding more straightforward changes in the snort configuration file. "output alert_unixsock: /var/run/snort_alert"

If Snort and Honeybrid are not on the same system, reading alerts from a socket involves network communication, which adds overhead. Another way is to manually spool the alert from the snort alert file, which has an overhead of opening a file and reading. When new alerts are appended to the file, we have to reread the file from the start to get the newly appended alerts. So, we opted for unsock implementation in our architecture as it is easy to implement a listening socket at Honeybrid for new alerts.

5.1.4 Honeybrid:

As discussed previously, Honeybrid is selected for its transparent traffic redirection and containment capabilities. It serves as the link between Snort and the TPOTCE Honeypots. Honeybrid is a popular choice for traffic redirection and analysis because it is easy to use and deploy. It is also very flexible and can redirect traffic to various honeypots, including low-interaction and high-interaction. Honeybrid also provides several features for analyzing attack traffic, such as recording sessions and generating reports. Honeybrid is an intermediary component that redirects suspicious traffic to the Honeynet and facilitates traffic containment. We are running Snort on the same VirtualBox container as Snort is running.

5.1.5 Configuring Netfilter rules:

The next step would be configuring the Netfilter firewall on the gateway so that Honeybrid can receive attack and honeypot traffic. This would be achieved using the following commands:

iptables -A PREROUTING -d 10.0.0.0/21 -j DNAT –to-destination 10.0.0.2 -m comment –comment "NAT rule for attack traffic to be sent to TPOT CE"

iptables -I FORWARD -d 10.0.0.3 -j QUEUE -m comment –comment "Incoming attack traffic"

iptables -I FORWARD -s 10.0.0.3 -j QUEUE -m comment –comment "Outgoing traffic from TPOT CE"

5.1.5.1 Mod_alert Module:

The architecture's modular design is highlighted by adding the "mod_alert" module. "mod_alert" assists in redirecting traffic to TPOTCE based on Snort alerts, providing a dynamic response to potential threats. The "mod_alert" module will listen on the socket for the alerts sent by the unsock mode of the snort. We return true in this module when alerts are sent for the matching destination IP address, protocol, and port. This further can be enhanced by matching the packet payload, which caused the alert in the Snort in the first place, for more accuracy. We leave that to future work. Thus, we can say "mod_alert" as a decision module in our architecture.

module "mod_alert" {
function = alert;
mode = back;
}

5.1.5.2 Target Rule:

In the below subsections, we see different Honeybrid modes and modules and how they are flexible in implementing any traffic decision and redirection for our architecture. As soon as the "mod_alert" module decides to redirect the traffic to Honeypot, target rules come into the picture to trigger the redirection module. Below is an example of an HTTP traffic target rule that listens for traffic on port 80 of the server, which sends all traffic to the server by default. With the backend rule, we trigger the Honeypot redirection, which uses the "mod_alert" module for redirection decisions and redirects traffic to Honeypot with IP in the backend rule.

target {
filter "dst net 10.0.0.2 and proto tcp and port 80";
backend 10.0.0.3 "mod_alert";
control "control";
}

The architecture suggests placing Honeypots and the network in separate VLANs to ensure network security and isolate the Honeynet from the actual network. VLANs are used to segregate network traffic. Honeybrid can support VLANs, which allows for the secure separation of the two networks.

By using different VLANs, we can have the same IP address as the actual server for the Honeypot. This further helps with overcoming the fingerprinting issue with our architecture that we have discussed in section 3.1.5

Here is an example of using VLAN for redirecting HTTP traffic to Honeypot with the same IP address:

target {
filter "dst net 10.0.0.2 and proto tcp and port 80";
backend 10.0.0.2 vlan 2 "mod_alert";
control "control";
}

5.1.5.3 Flexible Redirection:

The architecture supports flexible redirection to different Honeypots based on protocols such as FTP, HTTP, TCP, UDP, etc. We can create a new module for each Honeypot to detect protocol based on port filtering, for example, port 22 for SSH and port 443 for HTTPS. We can also detect protocols based on Snort alerts.

module "dionaea_back" {
function = dionaea;

mode = back;
}
target {
filter ”dst net 10.0.0.2 and port 22”;
backend 10.0.0.2 vlan 2 ”dionaea_back”;
control ”control”;
}

This flexibility allows for tailored redirection based on the nature of the detected alert.

5.1.5.4 Rate-Limiting and Control Rule Implementation:

For data control, Honeybrid provides different modes to detect traffic types, whether intranet traffic, internet traffic, or internal traffic between VLANs. Modules, together with the modes, provide a fine-tuning of data control.

- A ”control” mode to rate limit network packets based on source IP address. Packets are rejected after a source IP sends more than a given number of packets in a given period.
- The ”source_time” module only accepts packets from an IP in a specified period. A given IP address that tries to connect outside the allowed time frame will get rejected.
- ’internet’ mode to define a rule to control outgoing internet traffic initiated by honeypots
- ’intranet’ mode to define a rule to control intranet traffic initiated by honeypots
- ’internal’ mode is used to define an internal target; to catch an intra-LAN connection, it requires a separate VLAN.

These internal, intranet and internal modes help redirect traffic from a Honeypot to a different one. These modes help extend Honeynet based on VLANs; for example, we can have multiple TPOT CE on each on a single VLAN in the same network, and a single Honeybrid can be used to manage these TPOT CE. As discussed in section 3.1.8, our architecture also supports redirecting outgoing traffic from one Honeypot to another. This can be achieved by using the internal Honeybrid mode.

The Apache server and TPOTCE Honeypots are placed in the same VLAN for ease of implementation during the experiment. Using the above concepts, we show a sample target rule we used for the HTTP traffic in our architecture.

5.1.6 TPOTCE(HONEYNET):

This subsection shows the technical aspects of implementing the required Honeynet TPOT CE for our architecture. The Honeynet used for the implementation is TPOTCE. VirtualBox3, on which TPOT CE is running, is configured with the network interface of TPOT CE with IP 10.0.0.3 and configuration as per the table 5.3.

Table 5.3 Configuration of TPOT CE Honeynet

TPOT CE VirtualBox Configuration	
Operating System	Debian 11
RAM	8 GB
Processors	4
Storage	100GB
Network Adapter	NAT

We have installed TPOT in a standalone mode where all services and HOneypots are installed on a single machine. There is an option to install in distributed mode, where you must configure each Honeypot and system mapping.

Below is the list of the Honeypots that TPOT CE is hosting and the port mappings. TPOTCE helps detect the attack, which is redirected from the Apache server.

5.1.6.1 Configuring Snare and Tanner in TPOT CE:

The advantage of using the Snare tool is that we can clone any website to be served to different levels of depth by utilizing the cloner command, which helps Honeypot to be tightly coupled with the server. We cloned the website hosted in the Apache server to mimic the actual server.

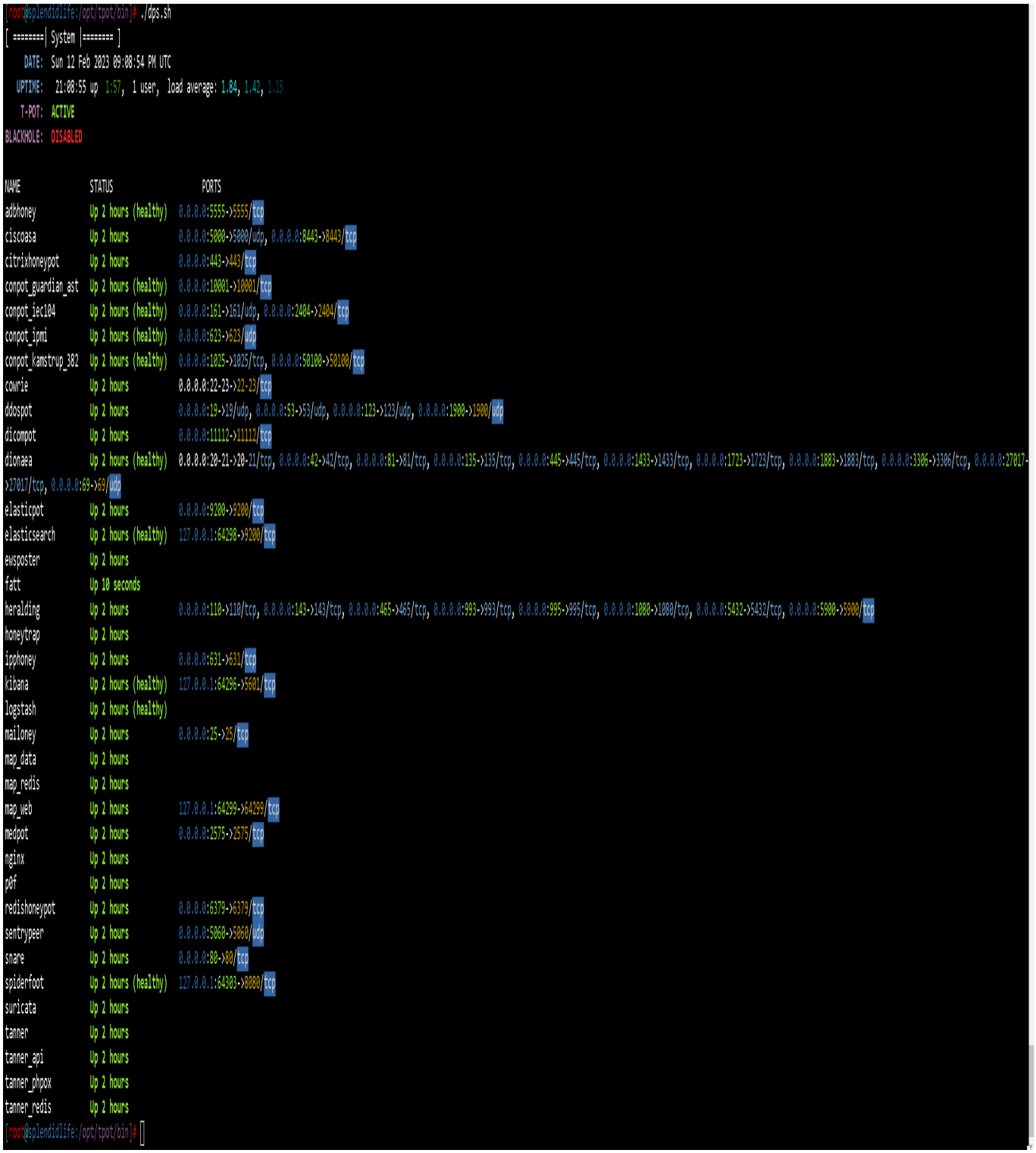

Figure 5.2 TPOT CE Honeypots and port configuration

Chapter 6

Experimental Evaluation

This chapter explains the experimental results based on the experiments conducted on the testbed implemented in the previous chapter. For evaluation, we used two types of datasets.

6.1 Evaluation through dataset:

1. A generated dataset is used to evaluate the true negative of the TPOTCE.
2. The CSE-CIC 2018 dataset was used to test the system's efficiency when input traffic is mixed with normal and malicious traffic in the real world instead of tool-generated traffic.

6.1.1 Tool generated traffic:

6.1.1.1 True Positive Simulation:

The primary goal of this experiment is to assess the accuracy of the Snare Honeypot in detecting HTTP-based attacks, particularly when integrated with the Snort intrusion detection system. The focus is on identifying true positives (genuine threats) and false positives (non-malicious activities incorrectly flagged as threats) at Honeypot(Snare).

6.1.1.2 Dataset Generation and Evaluation:

We will first simulate true positive detection of the current Honeypot architecture by redirecting the attack to Honeypot for attack detection. The experiment involves generating a synthetic dataset for testing purposes. This dataset simulates HTTP traffic with known attack payloads. The Snort IDS is configured with a specific rule set that includes signatures for SQL Injection and Cross-Site Scripting (XSS) attacks. These rules define patterns and characteristics associated with these attacks, enabling Snort to identify them in network traffic. The evaluation focuses on HTTP GET and POST requests, a common method for interacting with web servers.

1000 HTTP GET and POST requests are generated using a Python script with the Snort rule set for SQL injection and cross-site scripting attacks as input. The traffic is crafted so that Snort alerts are triggered, and in turn, a traffic redirection decision is made by the mod_alert module. Honeybrid will redirect the traffic to Tanner, which will evaluate the attack. All the attack traffic generated by the tool is detected at the attack at Snare, which is Tanner's brain. Using the Kibana tool, we can search and visualise data easily. Below are the logs of XSS attack detection at Tanner and Snare on the Kibana tool.

6.1.1.3 Evaluation of Snare using synthetic data:

Figure 6.1 Tanner XSS attack detection

We can also create dashboards on Kibana based on elastic data stored. Below is an example snapshot of Snare attacks with frequency.

Figure 6.2 Tanner top attacks

6.1.1.4 False Positive Simulation:

Now, to simulate the conditions of false positive detection in the current architecture, we forcefully redirect the attack-free normal traffic to Honeypot for detection. This is done by adding a custom rule set to snort rules,

which will trigger normal traffic as an alert. We created a simple ruleset based on the web page tree in our server and added them as the snort ruleset. A script generates false positive traffic based on the Snort rule set. This means the traffic is intentionally crafted to trigger Snort alerts, even though it is not genuinely malicious. All the attack-free traffic triggered alerts in Snort. Honeybrid, on the other hand, redirects this traffic to Honeypot. This simulates false positive alerted traffic at Snort, which Snare/Tanner should be detected as not an attack. Out of 500 such instances generated, all were seen as not an attack at Snare, i.e., no alert was generated.

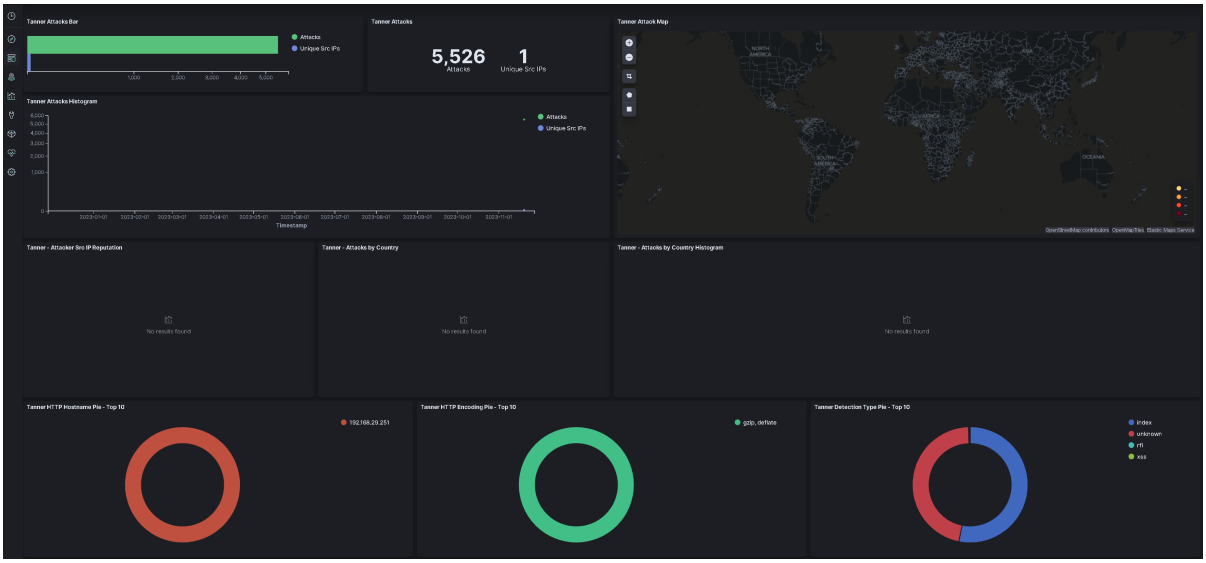

Figure 6.3 Tanner Dashboard visualising attack data

6.1.2 CSE-CIC-IDS2018 dataset:

Evaluating the hybrid Honeypot with a real-world dataset contributes to refining and improving its real-time detection capabilities. Insights gained from testing with actual traffic patterns can inform adjustments and enhancements to the system. Rather than using a synthetic dataset, as mentioned in the previous section, which is biased on the simulation environment and doesn't contain real-time false negatives, the CSE-CIC 2018 dataset comes into the picture. Also, we have tested only HTTP type traffic, specifically testing Snare Honeypot detection capability. As TPOT CE contains different types of Honeypots, we need a larger dataset to evaluate the overall performance of the architecture. The CSE-CIC-IDS2018 dataset is a public dataset that contains network traffic data collected from a simulated attack environment. The dataset contains a variety of attack types, including:

- Brute-force attacks: Brute-force attacks are attempts to gain unauthorised access to a system by trying many different passwords.
- Heartbleed attacks: Heartbleed attacks exploit a vulnerability in the OpenSSL cryptographic library, allowing attackers to steal sensitive data from servers.
- Botnet attacks: Botnet attacks are carried out by a network of compromised computers, known as a botnet.
- Denial-of-service (DoS) attacks: DoS attacks are attempts to make a system unavailable to legitimate users by overwhelming it with traffic.

- Distributed denial-of-service (DDoS) attacks: DDoS attacks are DoS attacks carried out by multiple attackers simultaneously.
- Web attacks: Web attacks are directed at web applications.
- Infiltration attacks: Infiltration attacks are attempts to gain unauthorised access to a system by exploiting vulnerabilities in the system's defences.

The CSE-CIC-IDS 2018 dataset contains a total of 16,233,002 instances gathered from 10 days of network traffic, of which 83% are regular instances and 17% are attack instances. This includes both training and test datasets.

Table 6.1 CSE-CIC-IDS 2018 Dataset traffic distribution

Network traffic distribution of CSE-CIC-IDS 2018 Dataset	
Attack type	Distribution (%)
Benign	83.070
Brute-force	2.347
Botnet	1.763
DoS	4.031
DDoS	7.786
Web attacks	0.006
Infiltration attacks	0.997

6.1.2.1 Modifying Dataset for Architecture:

Using tcpprep, we create a cache file based on the PCAP file. This cache file helps tcpreplay rewrite IP addresses and handle network configuration changes. Tcprewrite is used to modify the source and destination IP addresses in the PCAP file to match the desired IP address using the cached file. We used the Apache web server IP address as the destination address in PCAP. Tcpreplay command to replay the modified PCAP file to the

6.1.2.2 CSE CIC Dataset Challenges:

Many of the pcap files are corrupted. We used the pcapfix tool to fix them first. Some files that cannot be fixed using pcapfix are converted to a more robust pcappng format that may be less prone to corruption.

6.1.2.3 Evaluation of CSE CIC Dataset:

PulledPork v0.7.4, a Perl script application, is used to generate and manage the latest ruleset in Snort. As per the paper "Generating Labeled Training Datasets Towards Unified Network Intrusion Detection Systems" [29], the mismatch between actual attack numbers and detected attacks in Snort is mainly because the Snort ruleset cannot detect complete attacks in the dataset. Below are the alerts generated using the CSE-CIC 2018 dataset pcap file.

Table 6.2 Snort alert classification with PulledPork v0.7.4

Alerts generated by Snort using the CSE-CIC-IDS 2018 PCAP Dataset	
Attack type	Number of alerts
FTP & SSH Brute-force	2,82,784
DoS GoldenEye & Slowloris	8,185
DoS SlowHTTPTest	8139
DoS Hulk	205
DDoS LOIC HTTP & UDP	151
DDoS LOIC UDP	136
DDoS LOIC HOIC	329
Brute-force Web & XSS, SQL injection	630
Infiltration	4,196
Botnet	1,52,818

Upon detailed analysis of Snort alerts, Botnet attacks mainly contain HTTP traffic(Zeus and Ares Botnet), whereas Infiltration attacks mostly consist of Nmap scans.

Based on Snort's alert profile, we have added Honeybrid rules for FTP, HTTP, HTTPS, and SSH protocols.

Table 6.3 Protocol traffic and TPOT CE port mapping

Ports and Honeypot mapping in TPOT		
Protocol type	Port	Honeypot in T-pot
SSH	22	Cowrie Honeypot
FTP	20,21	Dionaea Honeypot
HTTP	80	Snare and Tanner
HTTPS	443	Citrix Honeypot

As for infiltration attacks, a separate Honeybrid rule is made to redirect the Snort-alerted traffic to TPOT CE. This is because we have added only one module, "mod_alert", which will redirect any alerted traffic to TPOT CE by changing the IP address. Hence, Other target rules of SSH, FTP, HTTP, and HTTPS will clash with the infiltration target rule. So, we have tested infiltration attacks separately with only the infiltration target rule. This can be overcome by creating a separate alert module for infiltration attacks, as it contains traffic over a wide range of ports and both TCP and UDP protocols. Below is the Honeybrid target rule for infiltration attacks. It contains all traffic going through the eth0 NAT interface. X.X.X.X/24 is the network with a subnet mask where the actual server exists, and Y.Y.Y.Y is the TPOT CE IP address.

```
    link "wan1" {
interface = "eth0";
```

filter = ”(tcp or udp) and dst net X.X.X.X/24”;
}

target default route via ”wan1” {
backend Y.Y.Y.Y ”mod_alert”;
internet ”control”;
}

All the Nmap scans are detected by the TPOT CE Sucricata tool. Below is the snapshot of Nmap scan logs in TPOT CE.

Figure 6.4 Nmap scan alerts in TPOT CE Suricata tool

The table in reference (Table 6.2) indicates that the number of alerts Snort detects for each attack is lower than the actual count of attacks in the dataset. This observation suggests that there were minimal false positives generated in Snort alerts. Additional testing of the same architecture with anomaly IDS/IPS is necessary to assess false positives thoroughly. Concerning false positives, due to the substantial volume of Snort alerts generated for the CSE-CIC-IDS dataset, manual investigation to verify the authenticity of these alerts and their classification as genuine attacks from the TPOT CE logs proved challenging. This challenge extends to the evaluation of true positives. Further correlation of alerts between Snort and TPOT CE will provide valuable insights for a more comprehensive evaluation of this architecture.

From the section 6.1.1, it is clear with experimentation of generated data that if an attack is flagged in the Snort alerts, it is subsequently identified as an attack or not with 100% accuracy in TPOT CE.

Below is the TPOT CE dashboard snapshot captured between the experiments.

Figure 6.5 TPOT CE Dashboard

6.2 Performance:

Our architecture's three major bottleneck components are Snort(IDS/IPS) and Honeybrid. Let us talk about the performance of each component concerning our architecture.

6.2.1 IDS/IPS Performace:

IDS or IPS is a fundamental component in any network, so delay caused by the alert generation or malicious traffic detection is negligible. Another performance aspect is sending alerts to the Honeybrid decision module "mod_alert" using sockets or spooling alert files. The delay is insignificant if IDS/IPS and Honeybrid are present on the same network. If IDS/IPS are far from Honeybrid, then socket communication delay can add up to 100 ms.

As per the experiment discussed in 6.1, the bottleneck regarding Snort is the alert generation. Snort alert generation mainly depends on the Snort ruleset. It is better to keep the ruleset as minimal and straightforward as possible to reduce the detection delay. Generally, we don't need to test false positives for the whole IDS/IPS ruleset; only newly added rules should be tested for false positives with this architecture.

6.2.2 Honeybrid Performace:

Honeybrid is the main bottleneck of our architecture. As per extensive testing done by Honeynet Organisation [27] on the Decision Module of Honeybrid, the hash module uses the hash function to compute payload checksums and accept packets that carry unknown checksums. The results for this module are stored in the file. On average, the hash module of Honeybrid took 35ms to decide on the fate of each packet, which is a pretty high value because it means no more than 29 packets can be handled per second. They are saving results to files that consumed most

of the total processing time and increased with Honeybrid recording more new payload checksums. Without this time-consuming task, the hash module took, on average, 0.6 ms to process packets, which means Honeybrid can currently handle a maximum of 1,500 packets per second. Module ”mod_alert”, implemented in our architecture, doesn’t use file reading or writing. So, on average, ”mod_alert” also will take 0.6 ms to process a packet. Thus, the capacity of Honeybrid using ”mod_alert” modules is 1500 packets per second.

Chapter 7

Conclusion

7.1 Thesis Summary:

Honeypots generally in a network are underutilised; current architecture can boost Honeypot utilisation and functionality. This paper aims to present a prototype that can be used to deploy and configure Honeypots hassle-free for IDS/IPS False positive detection. Honeypots add much value to our information about malicious activity, and combining it with Honeybrid provides better management. The synthetic dataset, which is generated, provided desirable results of the architecture as Snare and Tanner could detect the attack(True Positives) and false positives of IDS alerts with 100% accuracy. CSE-CIC-IDS 2018 dataset, we could not accurately evaluate the false positives as the Snort signature-based IDS used in the implementation generated very few false positives from the dataset. There is a further requirement for assessing the architecture with anomaly-based IDS, which can cause an ample amount of false positive alerts. Furthermore, evaluating the architecture with live traffic will help us determine redirection to Honeypots, making the real attackers interact more and help capture new attacks like remote file inclusion attacks.

7.2 Future Work:

Regarding future work, current architecture can be further optimised or explored at all three components: IDS/IPS, Honeybrid and TPOT CE. In future work, we plan to compare different IDS/IPS detection strategies by using various Honeypots and detection mechanisms apart from the ones used in this study. we plan to conduct a study on alert reduction in IDS for different intrusion detection algorithms, including anomaly-based and machine learning-based IDS/IPS.

As Honeybrid only supports 1500 connections per second to deploy the current architecture in a big-scale network, we need to enhance or optimise Honeybrid further. The current architecture supports only the redirection of malicious traffic to one Honeypot with a one-to-one mapping to alerts. Further scope of redirection of malicious traffic to multiple Honeypots must be explored. One way is redirecting traffic to Proxy Honeypots and replaying traffic to multiple Honeypots for verification from proxy honeypots. Further, there is a need to explore proxy services more as an alternative to Honeybrid.

Specifically, we aim to add more complicated emulators to our testing setup and evaluate the effectiveness of customised Honeypots and detection mechanisms in TPOTCE. We also plan to improve false positive mitigation methods to avoid choking actual users. To better automate the spawning of Honeypots inside TPOTCE. Further

feedback from the Honeypot evaluation of the malicious activity can be sent to IDS/IPS to improve the detection of attacks and reduce false positives.

Bibliography

[1] N. Provos, "Honeyd - Virtual Honeypot Daemon," Honeyd website, 2003. [Online]. Available: http://www.honeyd.org/background.php. [Accessed: 08-Mar-2023].

[2] Honeybrid, [Online]. Available: http://honeybrid.sourceforge.net/. [Accessed: 08-Mar-2023].

[3] M. Bailey, C. Evan, and D. Watson, "A hybrid honeypot architecture for scalable network monitoring," in *Recent Advances in Intrusion Detection*, 2004, pp. 242-261.

[4] W. Fan, Z. Du, D. Fernández, and X. Hui, "Dynamic Hybrid Honeypot System Based Transparent Traffic Redirection Mechanism," in *2016 IEEE Conference on Communications and Network Security (CNS)*, 2016, pp. 290-298.

[5] L. Spitzner, *Honeypots: Tracking Hackers*, Addison-Wesley Professional, 2002.

[6] Telekom Security. (2022). TPot is the all-in-one honeypot platform [GitHub repository]. Retrieved March 8, 2023, from https://github.com/telekom-security/tpotce

[7] Docker, Inc. (2022). Docker Compose [Documentation]. Retrieved March 8, 2023, from https://docs.docker.com/compose/

[8] Sourcefire, Inc. (2022). Snort - The World's Most Widely Used NIDS & IPS [Website]. Retrieved March 8, 2023, from https://www.snort.org/

[9] The Honeynet Project, "Honeywall CDROM," website, 2022. [Online]. Available: https://www.honeynet.org/projects/old/honeywall-cdrom/. [Accessed: 08-Mar-2023].

[10] S. Patton, W. Yurcik, and D. Doss, "An Achilles' Heel in Signature-Based IDS: Squealing False Positives in SNORT," 2001.

[11] S. Sudaharan and S. Dhammalapati, "Honeynet Clusters as an Early Warning System for Production Networks," 2005.

[12] M. Vrable, J. Ma, J. Chen, and J. Moore, "Scalability, fidelity, and containment in the Potemkin virtual honeyfarm," 2005.

[13] K. G. Anagnostakis, S. Sidiroglou, P. Akritidis, K. Xinidis, E. Markatos, and A. D. Keromytis, "Detecting Targeted Attacks Using Shadow Honeypots," 2005.

[14] C. Leita and D. Dacier, "Automatic Handling of Protocol Dependencies and Reaction to 0-Day Attacks with ScriptGen Based Honeypots," 2006.

[15] R. G. Berthier and M. Cukier, "Advanced honeypot architecture for network threats quantification," 2009.

[16] I. Alberdi, E. Alata, P. Owezarski, V. Nicomette, and M. Kaˆaniche, "Shark: Spy Honeypot with Advanced Redirection Kit," 2007.

[17] C. Kreibich, "Honeycomb," International Computer Science Institute, [Online]. Available: `http://www.icir.org/christian/honeycomb/index.html`. Accessed on March 8, 2023.

[18] P. Sokol, J. Míšek, and M. Husák, "Honeypots and honeynets: issues of privacy," 2017.

[19] Wikipedia contributors, "DMZ (computing)," Wikipedia, The Free Encyclopedia, last updated February 15, 2023, accessed March 8, 2023, https://en.wikipedia.org/wiki/DMZ_(computing).

[20] Netfilter contributors, "Netfilter," Netfilter.org, accessed March 8, 2023, https://www.netfilter.org/.

[21] The Honeynet Project, "Glastopf - A Dynamic, Low-Interaction Web Application Honeypot," *The Honeynet Project*, 2012. [Online]. Available: `https://www.honeynet.org/projects/glastopf/`. [Accessed: 08-Mar-2023].

[22] Xuxian Jiang and Dongyan Xu, "Collapsar: A VM-Based Architecture for Network Attack Detention Center," in *Proceedings of the 2004 ACM Workshop on Rapid Malcode*, Washington, DC, USA, 2004, pp. 1-9.

[23] Canadian Institute for Cybersecurity, University of New Brunswick. (2018). CSE-CIC-IDS2018 dataset. [Online]. Available: `https://www.unb.ca/cic/datasets/ids-2018.html`. [Accessed: 08-Mar-2023].

[24] Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and ROC curves. In Proceedings of the 23rd International Conference on Machine learning (ICML '06), pages 233–240, New York, NY, USA. ACM.

[25] Honeypots and Honeytraps: Unveiling Cyber Intrusion Detection Tactics. https://f60host.com/support/honeypots-honeytraps-cyber-intrusion-detection/

[26] Host-Based vs Network-Based Intrusion Detection System (IDS) - Logix Consulting Managed IT Support Services Seattle. https://logixconsulting.com/2022/05/03/host-based-vs-network-based-intrusion-detection-system-ids/

[27] https://honeynet.org/2009/08/07/honeybrid-testing/

[28] 13th USENIX Security Symposium — A Virtual Honeypot Framework. https://www.usenix.org/legacy/publications/library/proceedings/sec04/tech/full_papers/provos/provos_html/

[29] Generating Labeled Training Datasets Towards Unified Network Intrusion Detection Systems — . https://kyushu-u.pure.elsevier.com/ja/publications/generating-labeled-training-datasets-towards-unified-network-intr