Glutton: A Highly Sensitive, Protocol-Agnostic Honeypot

Muhammad Bilal Arif[1], Lukas Rist[1], and Yumna Ghazi[2]

[1]The Honeynet Project
[2]Block360 LLC dba Alphabase®

April 15, 2025

Abstract

This paper introduces Glutton, a highly sensitive, protocol-agnostic honeypot designed to detect network attacks that traditional honeypots can miss. Glutton sets itself apart by its unique ability to capture usually overlooked interactions, such as low-volume scans on non-standard ports and incomplete or incorrect usage of network protocols. The system leverages IPTables TPROXY for transparent traffic redirection, implements a dynamic rule engine for transport layer routing, and provides a flexible platform for implementing protocol-specific handlers. Network packets with an unknown application layer protocol are captured, and attempts are made to solicit an interaction from the adversary. Glutton's open architecture, coverage of all ports, and transport layers provide a unique extension for conventional threat intelligence solutions.

1 Introduction

Modern attackers increasingly rely on stealthy techniques like low-volume scans, partial protocol handshakes, and subtle behavioral anomalies to evade detection by conventional honeypots [1]. These systems often fail to capture such activities due to rigid protocol emulation, incomplete logging, or reliance on predefined attack signatures [2]. This paper introduces Glutton, a novel protocol-agnostic, server-side [3], low-interaction [4] honeypot [5] designed to address these limitations.

Glutton distinguishes itself through its port-agnostic architecture that accepts traffic across all ports and protocols. This approach enables it to capture a wide range of interactions without requiring specific protocol implementation for each service. By leveraging IPTables TPROXY for transparent traffic redirection, Glutton intercepts and logs connection attempts to all ports, providing visibility into reconnaissance activities that might otherwise go undetected.

While Glutton shares some limitations with other low-interaction honeypots, such as limited depth of protocol emulation, it offers the following advantages:

- **Protocol-agnosticism:** Instead of fully emulating each protocol, Glutton uses configurable rules and generic handlers to process any TCP/UDP-based interaction on all ports.
- **Detailed logging:** Glutton records all connections, including metadata and payloads, to preserve even partial interactions for further analysis.
- **Extensibility:** Its flexible mapping of protocol handlers and configurable rules allows it to quickly adapt to support new protocols without the need for extensive architectural changes.

Unlike specialized honeypots that focus on depth in a single protocol, Glutton serves as a breadth-oriented system for capturing various attack patterns across the entire port space. While it does not support interactivity with protocols, it excels at capturing connection attempts and their payloads, providing insights into early-stage reconnaissance and attack behaviors.

Related Work

Generic, highly customizable honeypot systems have been proposed and implemented during deception technologies' early inception [6], recognizing the need for quick adoption to capture emerging threats in their initial exploitation. Previous honeypot architectures used advanced Linux networking to enable the capture of network traffic on arbitrary ports. Tiny Honeypot [7] utilized the IPTables REDIRECT rule to enable coverage for all open ports. Honeytrap [8] uses the IPTables target NFQUEUE to intercept network traffic and spawn a listener on connection attempts on an arbitrary port.

Several tools have addressed the need for multi-protocol monitoring. T-Pot [9] integrates over 20 distinct honeypot systems, enabling cross-correlation of threat data across various protocols. Nepenths [10] introduces a flexible, extensible architecture, reducing the work required to modify and add application-level protocols to the honeypot. Collectively, these practical implementations validate the importance of protocol-agnostic honeypot frameworks capable of receiving connection requests on arbitrary ports. Such systems provide more visibility into modern attackers' diverse tactics, enabling more effective defense measures.

Architecture

Glutton's architecture prioritizes sensitivity over protocol-specific depth, enabling the capture of network interactions across all ports. The system consists of three core components: traffic redirection through IPTables TPROXY, a dynamic Rule Engine, and protocol-specific handlers.

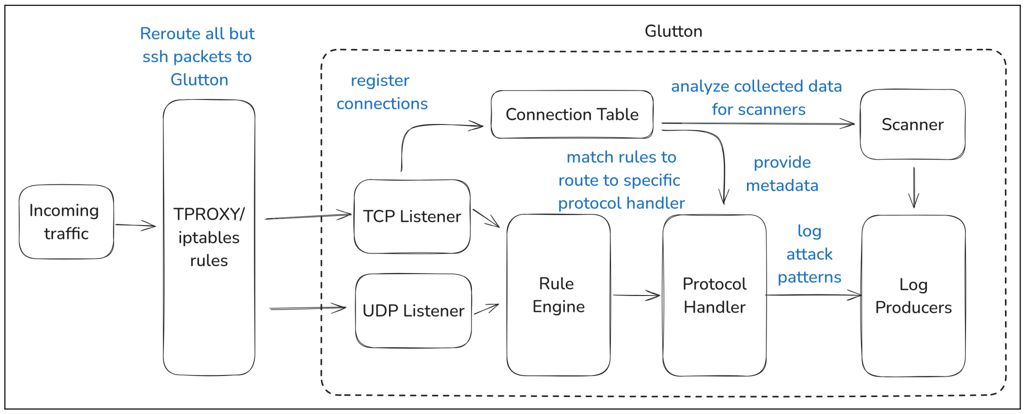

Figure 1: High-Level Architecture of Glutton

Glutton uses IPTables TPROXY to intercept inbound TCP and UDP traffic at the network layer. TPROXY captures packets directed to a specific destination and reroutes them to a proxy server without modifying the network layer. By default, Glutton listens locally on TCP port 5000 and UDP port 5001, preserving the original destination port information. Port 22 (SSH) is also excluded to ensure uninterrupted system access. The following IPTables rules enable this functionality:

```
iptables -t mangle -I PREROUTING -p tcp ! --dport 22 -m state !
--state ESTABLISHED,RELATED -j TPROXY --on-port 5000 --on-ip 127.0.0.1

iptables -t mangle -I PREROUTING -p udp ! --dport 22 -m state !
--state ESTABLISHED,RELATED -j TPROXY --on-port 5001 --on-ip 127.0.0.1
```

As connections arrive, they are registered in a centralized Connection Table that stores detailed metadata, including timestamp, destination port number, and matched rule. Combined with logged payloads, this data ensures that even partial or incomplete connections are preserved for analysis.

The Rule Engine processes these connections using BPF (Berkeley Packet Filter) syntax to define flexible traffic routing rules at the transport layer. It determines how incoming TCP and UDP traffic is directed to appropriate protocol handlers.

```
rules:
  - match: tcp dst port 23 or port 2323 or port 23231 # BPF syntax
    type: conn_handler
    target: telnet
  - match: tcp dst port 25
    type: conn_handler
    target: smtp
  - match: tcp dst port 21
    type: conn_handler
    target: ftp
```

Once the rules engine routes a connection, Protocol Handlers analyze the application-layer data, extracting details such as HTTP headers or FTP commands for more fine-grained routing. The system logs all interactions regardless of whether they represent complete protocol exchanges or partial handshakes. A scanner module checks incoming IP addresses against known scanning sources and sets the scanner name in the event logs for added context.

The Log Producer stores detailed metadata for each connection, including timestamps, source/destination IP and port pairs, protocol type, unique connection identifiers, and any captured payloads. It supports multiple output channels, including local file storage in structured formats (e.g., JSON), integration with HPFeeds for real-time threat intelligence sharing, and direct forwarding to external logging platforms. The system automatically sanitizes sensitive internal IPs and provides structured output suitable for integration with visualization tools like the ELK Stack or the Honeynet Project's Ochi visualization tool.

Log Producers follow a generic interface, allowing additional producers to implement it easily. This follows the project's general vision of being quickly adapted to emerging events.

Limitations and Future Work

While Glutton demonstrates the value of a protocol-agnostic honeypot architecture, it shares limitations with other low-interaction honeypots. The current design excels at capturing initial connection attempts and early-stage reconnaissance but has limited capabilities for extended attacker engagement. Enhancing protocol handlers with more sophisticated service emulation and integrating parsing frameworks like Spicy would enable the development of handlers at a scale that can better engage attackers and extract more detailed interaction patterns.

The current protocol handlers are limited in scope and session duration. Future work should focus on developing more interactive handlers to maintain longer sessions with attackers, potentially using Large Language Models (LLMs) for dynamic, contextual response generation.

Integration with external threat intelligence platforms (e.g., Open Threat Exchange or TAXII-based systems) would allow sharing of captured data in standardized formats so that the collected information can be made openly available to help the cybersecurity community. These improvements would enhance Glutton's ability to collect and provide rich data about attack patterns and techniques, providing security analysts with deeper insights into attacker behavior.

References

[1] Javadpour, A., Ja'fari, F., Taleb, T., Shojafar, M., & Benza"id, C. (2024). A comprehensive survey on cyber deception techniques. *Computers & Security*, 140, 103792.

[2] Dornseif, M., Holz, T., & Müller, S. (2005). Honeypots and Limitations of Deception.

[3] Seifert, C., Welch, I., & Komisarczuk, P. (2007). Honeyc—the low-interaction client honeypot. *NZCSRCS*, 6, 48.

[4] Mokube, I., & Adams, M. (2007). Honeypots: concepts, approaches, and challenges. *ACM Southeast Conference*, 321-326.

[5] Spitzner, L. (2003). The honeynet project: Trapping the hackers. *IEEE Security & Privacy*, 1(2), 15-23.

[6] Provos, N. (2004). A Virtual Honeypot Framework. *USENIX Security Symposium*, 173(2004), 1-14.

[7] George Bakos. (2025). Tiny honeypot. `https://github.com/tiny-honeypot/thp`

[8] Tillmann Werner. (2025). Honeytrap. `https://github.com/tillmannw/honeytrap`

[9] Deutsche Telekom AG. (2025). T-Pot honeypot platform. `https://github.com/telekom-security/tpotce`

[10] Baecher, P., Koetter, M., Holz, T., Dornseif, M., & Freiling, F. (2006). The nepenthes platform. *RAID 2006*, 165–184.