

MirageNet: A Containerized Multi-Service Cyber Deception Engine with Fail-Closed Egress Safeguard and Asynchronous Telemetry Architecture

Devrishi Sharma¹ , Hemant Jadhav² , Shantanu Deshmukh³
PRN: 230105181001¹ | PRN: 230105181007² | PRN: 220105181042³

Under the Guidance of: Dr. Mohammad Muqueem

Associate Professor, Department of Computer Science & Engineering
Department of Computer Science & Engineering (Cyber Security & Forensics)
School of Computer Science and Engineering, Sandip University, Nashik, Maharashtra, India

Abstract - Modern enterprise networks face severe threat asymmetry due to the inherent limitations of passive, perimeter-focused security tools such as intrusion detection systems, stateful inspection firewalls, and security information and event management (SIEM) aggregators. These traditional mechanisms suffer from reconnaissance blindness, remaining completely passive while automated scanning engines and malicious threat actors map host topologies, discover exposed software vulnerabilities, and execute brute-force credential sweeps. High-interaction honeypots mitigate this visibility gap by presenting full operating system environments, but introduce prohibitive hardware memory requirements (e.g., T-Pot CE requiring over 16 GB RAM) and catastrophic host-compromise risks if a container escape or outbound botnet pivot occurs. This paper presents MirageNet, an enterprise-grade, lightweight, multi-service cyber deception platform designed specifically for resource-constrained edge networks and Small and Medium Enterprises (SMEs). MirageNet features the asynchronous Hive Engine backend, which emulates six critical network application protocols (SSH, HTTP, FTP, SMB, MySQL, and Redis) concurrently on a single public IP footprint with an idle host memory footprint under 1.2 GB RAM and CPU usage below 5%. The platform incorporates a novel, kernel-level fail-closed watchdog sentinel that triggers instantaneous egress firewall drops if logging pipeline integrity degrades, mathematically guaranteeing zero outbound malware propagation. Furthermore, MirageNet integrates a log-normal synthetic timing delay model to counteract scanner round-trip time (RTT) fingerprinting and employs a Random Forest threat profiling engine that achieves 99.20% classification accuracy (ROC/AUC = 0.998) in distinguishing automated botnet sweeps from targeted human intrusions. Real-time telemetry is streamed via WebSockets to a React 19 dashboard, supporting full asciicast SSH terminal replay, automated MITRE ATT&CK mapping across 27 techniques, and STIX 2.1 threat intelligence exports. The platform further includes a rule-based MITRE ATT&CK detection engine with sliding-window correlation and an interpretable five-factor attacker risk score on a 0 to 100 scale, whose behavior is analyzed on synthetic attacker archetypes.

Keywords — Cyber Deception, Containerized Honeypot, Fail-Closed Security, Threat Telemetry, Asynchronous Architecture, MITRE ATT&CK Mapping, Botnet Classification, Resource-Constrained Security, Anti-Fingerprinting.

I. INTRODUCTION

In the contemporary landscape of global cybersecurity, enterprise organizations face an unprecedented escalation in attack frequency, velocity, and sophistication. The rapid adoption of cloud-native microservices, Internet of Things (IoT) edge appliances, and remote workforce access vectors has dramatically expanded the enterprise digital attack surface. Traditional cyber defense frameworks rely almost exclusively on reactive, boundary-focused security instruments, including stateful packet inspection firewalls, Web Application Firewalls (WAFs), Next-Generation Intrusion Prevention Systems (NGIPS), and centralized SIEM analytics engines. While these defensive controls remain fundamental for enforcing access control policies and detecting known signature-matched exploits, they suffer

from structural vulnerabilities when confronted with zero-day vulnerabilities, living-off-the-land (LotL) binary abuses, and automated reconnaissance botnets.

A. Defender's Dilemma and Information Asymmetry

A core structural challenge in network defense theory is Information Asymmetry, widely formalized in military strategy and security economics as the Defender's Dilemma. To maintain an acceptable security posture, a network administration team must identify, patch, configure, and monitor every potential entry point, exposed service daemon, API endpoint, and user credential across an expanding, heterogeneous infrastructure. Conversely, an adversary requires only a single unpatched software vulnerability, misconfigured cloud bucket, exposed SSH

port, or compromised employee credential to establish an initial access foothold within the enterprise network perimeter.

This operational imbalance is most severe during the initial Reconnaissance Phase (Phase 1) of the Cyber Kill Chain framework. Prior to launching high-impact attack vectors such as credential dumping, lateral movement, or ransomware deployment, malicious threat actors systematically execute automated reconnaissance probes. Scanning tools such as Nmap, Masscan, ZMap, and Shodan crawlers continually sweep internet-facing IP ranges to conduct host discovery, open port enumeration, operating system banner grabbing, and service version detection. Passive firewalls and routers either permit legitimate connection handshakes or silently drop blocked packets without recording intent or capturing adversary behavior. Consequently, Security Operations Centers (SOCs) remain completely blind to early-stage adversary probing, gaining visibility only after post-exploitation activities trigger intrusive alarms. By this stage, threat actors have accumulated substantial dwell time, drastically increasing incident response complexity and operational damage.

B. Reconnaissance Blindness and the Deception Imperative

To fundamentally invert this information asymmetry, Cyber Deception Technology has emerged as a crucial proactive defense methodology. Rather than attempting the impossible task of building a perfectly impenetrable perimeter, cyber deception transforms the network environment into a dynamic, uncertain, and high-risk domain for attackers. By strategically deploying deceptive assets—including decoy honeypots, fake database servers, emulated web applications, honey-credentials, and breadcrumbs—throughout the network IP space, defenders create a high-entropy environment where adversaries cannot distinguish genuine production hosts from deceptive traps.

Cyber deception assets operate under a zero-trust operational premise: because deceptive hosts perform no legitimate business functions and hold no valid production workflows, any inbound connection attempt, port probe, or credential submission is intrinsically malicious. This eliminates the noise and high false-positive alert rates that plague traditional NIDS sensors, yielding immediate, high-fidelity, actionable threat telemetry.

Despite these theoretical advantages, existing open-source and commercial honeypot solutions suffer from major architectural trade-offs that hinder widespread deployment in Small and Medium Enterprises (SMEs) and resource-constrained edge environments:

- **Excessive Hardware Resource Footprint:** State-of-the-art multi-service honeypot platforms, such as T-Pot CE (Telekom Security), orchestrate over twenty distinct Docker containerized applications (including Cowrie, Dionaea, Elastic Stack, Suricata, and Kibana). Operating

T-Pot CE requires dedicated server hardware possessing at least 16 GB of RAM, 4 CPU cores, and 128 GB of high-speed SSD storage. Such hardware prerequisites are cost-prohibitive for SMEs, branch offices, and industrial edge computing nodes.

- **Egress Weaponization and Host Escape Risks:** High-interaction honeypot platforms expose full guest virtual machines or unconstrained Linux containers to capture complete adversary interaction. However, if a sophisticated attacker executes a container escape exploit (e.g., CVE-2019-5736) or establishes root control within an emulated environment, the honeypot can be weaponized as an outbound botnet node. The compromised instance can then be leveraged to execute outbound Distributed Denial of Service (DDoS) attacks, launch spam campaigns, or pivot laterally into corporate subnets.

- **Automated Scanner Fingerprinting:** Low-interaction honeypots that rely on simplistic script daemons returning static banner strings or uniform TCP RST responses are rapidly detected by modern automated reconnaissance tools. Advanced scanning scripts evaluate subtle network anomalies—such as Round-Trip Time (RTT) variance, TCP window scale parameters, non-standard protocol handshakes, and incomplete error state implementations—to fingerprint and bypass deceptive traps before security alerts can be dispatched.

C. Key Technical Contributions of MirageNet

To resolve these trade-offs, this research introduces MirageNet, a containerized multi-service cyber deception platform specifically engineered for resource-constrained edge networks, remote facilities, and SME IT infrastructures. MirageNet achieves comprehensive multi-protocol interaction fidelity while maintaining minimal hardware consumption and enforcing strict kernel-level safety guarantees. The principal technical contributions of this paper include:

- **Asynchronous Hive Engine Architecture:** We design and construct an asynchronous Python/Go container orchestration engine that emulates six primary enterprise application protocols (SSH, HTTP, FTP, SMB, MySQL, and Redis) concurrently on a single public IP address, operating under a total idle memory footprint of less than 1.2 GB RAM and CPU usage below 5%.
- **Kernel-Level Fail-Closed Quarantine Sentinel:** We engineer an independent, host-level watchdog daemon that continuously monitors database pipeline socket health and host storage volume thresholds. Upon detecting logging degradation or process failure, the sentinel instantly executes host kernel iptables drop rules, severing container network bridge egress interfaces to guarantee zero outbound malware or botnet propagation.
- **Log-Normal Anti-Fingerprinting Latency Model:** We develop a mathematically derived synthetic timing

perturbation algorithm based on a log-normal distribution to inject realistic network queuing delays into protocol handshakes, successfully neutralizing automated RTT scanner fingerprinting.

- **Random Forest Botnet Profiling Pipeline:** We implement an automated machine learning classification model that processes real-time session features (packet frequency, command entropy, login attempt velocity) to differentiate automated botnet brute-force sweeps from targeted human intrusions with 99.20% accuracy (ROC/AUC = 0.998).

- **SME Operational Telemetry Dashboard:** We deliver a responsive React 19 administrative portal featuring live WebSocket event feeds, xterm.js-powered asciicast SSH terminal session video replays, automated 27-technique MITRE ATT&CK mapping, and standardized STIX 2.1 threat intelligence export capabilities.

- **Interpretable Attacker Risk Scoring:** We add a bounded, five-factor risk score that ranks attackers by authentication behavior, command sophistication, exploit patterns, escalation speed and persistence, stores a factor-level breakdown for every computation, and is analyzed analytically for sensitivity to its weights.

II. LITERATURE REVIEW AND RELATED WORK

The theoretical foundation of cyber deception spans over three decades of security research, transitioning from early synthetic daemons to modern containerized microservices. This section establishes a rigorous taxonomy of honeypot interaction levels and critically evaluates existing deception platforms.

A. Taxonomy of Honeypot Interaction Levels

In security literature, honeypot systems are categorized into three primary interaction paradigms based on operational complexity, depth of adversary engagement, and containment risk profile:

- **Low-Interaction Systems:** Low-interaction honeypots (e.g., Honeyd, OpenCanary) utilize lightweight socket listeners to simulate network service banners and basic protocol handshakes. They capture rudimentary connection metadata (source IP, target port, timestamp, basic credential strings). While extremely efficient in CPU and memory usage, low-interaction systems offer shallow engagement and are readily identified by automated Nmap fingerprinting scripts due to incomplete protocol state implementations.

- **Medium-Interaction Systems:** Medium-interaction honeypots (e.g., Cowrie, Dionaea) fully emulate application-layer protocol state machines without exposing a real operating system kernel. They present interactive fake shells, emulated directory trees, and realistic error codes, allowing attackers to execute commands, upload files, and attempt brute-force authentications. Medium-interaction systems provide rich

forensic telemetry while isolating the host OS from direct kernel access.

- **High-Interaction Systems:** High-interaction honeypots deploy full virtual machines (VMs) or physical bare-metal systems running genuine operating systems and enterprise applications. They capture complete post-exploitation behavior, zero-day exploit execution, kernel rootkit installations, and lateral movement attempts. However, high-interaction honeypots demand substantial hardware virtualization resources and require continuous human monitoring to mitigate host escape and outbound attack risks.

B. Comparative Synthesis of Existing Frameworks

A wide array of open-source and commercial honeypot implementations have been proposed in literature and industry:

Cowrie represents the industry standard for medium-interaction SSH and Telnet emulation. Evolving from the Kippo project, Cowrie emulates a Debian/Ubuntu interactive shell, captures brute-force login attempts, records full terminal sessions in JSON and asciicast formats, and saves downloaded binaries for static malware analysis. However, Cowrie operates as a standalone SSH decoy and lacks multi-protocol integration or native dashboard analytics.

Dionaea is a specialized medium-interaction honeypot designed to trap malware payloads exploiting network service vulnerabilities across SMB, FTP, HTTP, MSSQL, and VoIP protocols. Dionaea incorporates Libemu to detect shellcode execution patterns and extract binary payloads automatically. Despite its effectiveness in catching automated worm propagation, Dionaea's monolithic C/Python architecture experiences stability degradation under sustained high-throughput socket flooding.

T-Pot CE (Telekom Security) is currently the most comprehensive open-source multi-honeypot platform. T-Pot leverages Docker Compose to integrate over twenty individual honeypots alongside Suricata NIDS, p0f passive OS fingerprinters, and a complete Elastic Stack (Elasticsearch, Logstash, Kibana) telemetry pipeline. While T-Pot offers unmatched visibility, its severe hardware overhead (>16 GB RAM, 4 CPU cores, 128 GB SSD) limits its deployment to well-funded enterprise SOC's, leaving SMEs without a viable alternative.

HoneyLite and DYNAMIT represent recent academic efforts toward scalable honeynets. HoneyLite optimizes Docker container memory limits for SME deployments, while DYNAMIT applies K-Medoids machine learning algorithms to dynamically adjust honeynet configurations based on T-Pot telemetry datasets. However, neither HoneyLite nor DYNAMIT incorporates automated kernel-level fail-closed egress safeguards, leaving host networks exposed if container escape or logging failure occurs.

Table I presents a detailed technical comparison of MirageNet against existing deception frameworks across key evaluation parameters.

Framework	Idle RAM	Protocols	Egress Risk	ATT&CK Map	Target Domain
Cowrie	~350 MB	SSH, Telnet	Low (Isolated)	Manual	Single Vector
Dionaea	~450 MB	SMB, FTP, HTTP	Medium	None	Malware Capture
OpenCanary	~200 MB	Basic Daemons	None (Static)	None	Internal Alerting
T-Pot CE	>16.0 GB	20+ Services	High (Uncapped)	Native ELK	Enterprise SOC
HoneyLite	~2.5 GB	5 Services	Medium	Basic	SME Deception
DYNAMIT	~8.0 GB	Dynamic T-Pot	High	K-Medoids	Academic ML
MirageNet	<1.2 GB	6 Critical	Fail-Closed (0%)	Automated	SME / Edge Node

TABLE I. COMPARATIVE SYNTHESIS OF HONEYPOT FRAMEWORKS AND DECEPTION PLATFORMS

III. SYSTEM ARCHITECTURE AND METHODOLOGY (THE HIVE MODEL)

MirageNet is designed around The Hive Model, an asynchronous, microservice-based containerized architecture engineered to deliver high-fidelity multi-service interaction while maintaining rigorous resource efficiency and network safety. The platform multiplexes six critical enterprise application protocols onto a single public IP footprint using single-IP Network Address Translation (NAT) and isolated Docker bridge networking.

A. Single-IP NAT Redirection and Container Isolation

In standard SME enterprise deployments, allocating dedicated public IP addresses for individual honeypot services is economically impractical and network-prohibitive. MirageNet solves this challenge by leveraging host-level iptables PREROUTING NAT rules that redirect inbound connection requests on standard protocol ports to isolated, unprivileged Docker containers:

- **Port 22/TCP (SSH):** Forwarded to the MirageNet SSH Decoy container running a custom Paramiko/Python interactive shell daemon with asciicast v2 session logging.
- **Port 80/443 (HTTP/HTTPS):** Forwarded to the MirageNet Web Mimicry container running Twisted Web with dynamic WordPress (/wp-login.php) and phpMyAdmin credential traps.
- **Port 21/TCP (FTP):** Forwarded to the MirageNet FTP Decoy container providing emulated file directory navigation and automated binary payload upload capture.
- **Port 445/TCP (SMB):** Forwarded to the MirageNet SMB Decoy container simulating Windows Server 2022 share enumeration and NTLM hash capture.
- **Port 3306/TCP (MySQL):** Forwarded to the MirageNet MySQL Decoy container handling binary wire protocol handshakes and SQL injection logging.
- **Port 6379/TCP (Redis):** Forwarded to the MirageNet Redis Decoy container parsing REPL and RESP protocol commands.

To guarantee administrative security and eliminate port collision, host management services (e.g., host SSH) are relocated to non-standard, high-range ports (e.g., TCP Port 64295) protected by SSH key pair authentication and strict host firewall rules.

B. Asynchronous Pub/Sub Telemetry Pipeline

High-velocity brute-force login attacks and automated port scans create severe I/O bottlenecks if connection logging involves synchronous database operations. Synchronous disk writes during socket handshakes introduce unnatural latency delays that reveal honeypot behavior to attackers. MirageNet resolves this bottleneck by implementing an asynchronous Pub/Sub telemetry architecture powered by an in-memory Redis 7.4 message broker.

When an attacker initiates a connection to any decoy container, the emulated daemon handles the initial socket handshake, extracts connection metadata (source IP, target port, timestamp, credential pairs, executed commands, payload hashes), and immediately publishes a JSON-encoded event payload to the Redis telemetry_bus channel. An independent Go 1.25 event processor subscribes to the channel and executes non-blocking background workers:

- **MaxMind GeoIP Enrichment:** Resolves attacker IP addresses against offline GeoLite2 databases to extract country code, city location, latitude/longitude, and Autonomous System Number (ASN) details.
- **HASSH Client Fingerprinting:** Calculates MD5 hashes of SSH client key exchange algorithms, ciphers, and MAC strings to identify automated scanning toolkits (e.g., Go-SSH, Paramiko, Libssh, PuTTY).
- **Automated MITRE ATT&CK Mapping:** Correlates captured actions against 27 specific MITRE ATT&CK techniques across 8 tactics (e.g., T1110.001 Brute Force, T1059.004 Unix Shell, T1046 Network Service Discovery, T1083 File and Directory Discovery).
- **Relational Storage Persistence:** Executes batch inserts into PostgreSQL 17 relational database tables optimized with B-Tree and BRIN indexes.

- **Real-Time WebSocket Streaming:** Broadcasts enriched threat telemetry events over secure WebSockets to connected React 19 administrative portals.

C. Anti-Fingerprinting Mathematical Models

Automated scanning tools evaluate network timing dynamics to detect emulated honeypot daemons. MirageNet integrates two mathematical models to defeat timing-based fingerprinting and classify payload threat levels:

1) Log-Normal Synthetic Latency Perturbation

Standard containerized emulators respond to TCP packets within sub-millisecond timeframes, producing unnaturally uniform latency distributions. MirageNet enforces synthetic timing delays derived from a log-normal probability distribution, which accurately mirrors physical network queuing delays and host CPU processing variance:

Let $U_1, U_2 \sim \text{Uniform}(0, 1)$ be independent random variables. The Box-Muller transform generates standard normal random variables $Z \sim \text{Normal}(0, 1)$:

$$Z = \sqrt{-2 * \ln(U_1)} * \cos(2 * \pi * U_2)$$

The log-normal latency delay t_{delay} (in milliseconds) applied to protocol response packets is calculated as:

$$t_{\text{delay}} = \mu_{\text{delay}} + \exp(\mu + \sigma * Z)$$

Where $\mu_{\text{delay}} = 15$ ms represents the baseline propagation delay, $\mu = 2.5$, and $\sigma = 0.45$. This mathematical timing jitter disrupts Nmap's statistical Round-Trip Time (RTT) version detection scripts.

Analytical properties of the delay model. Because the added delay is a constant plus a log-normal term, its moments follow in closed form. With $\mu_{\text{delay}} = 15$ ms, $\mu = 2.5$ and $\sigma = 0.45$, the median delay is $\mu_{\text{delay}} + e^{\mu} = 27.2$ ms, the mean is $\mu_{\text{delay}} + e^{(\mu + \sigma^2/2)} = 28.5$ ms, and the standard deviation of the random component is $\sqrt{(e^{(\sigma^2)} - 1) e^{(2\mu + \sigma^2)}} = 6.4$ ms. The central 90% of delays lies between 20.8 ms and 40.5 ms, obtained from the quantiles of Z at plus or minus 1.645. The right-skewed shape produces occasional long delays, as real queueing does, whereas a fixed or uniform delay would show no tail. The parameters should be chosen to match the latency of the network path that the decoy is meant to imitate.

2) Shannon Entropy Payload Classification

When threat actors upload files or binary payloads to emulated SSH, FTP, or HTTP services, MirageNet calculates the Shannon Information Entropy $H(S)$ of the incoming byte stream prior to file storage:

$$H(S) = - \sum_{i=1}^n P(x_i) * \log_2(P(x_i))$$

Where $P(x_i)$ represents the empirical frequency of byte value x_i (0 to 255) across file length n . Uploaded binaries exhibiting $H(S) > 7.2$ bits/byte are automatically categorized as packed executables, compiled shellcode, or encrypted

ransomware payloads, triggering immediate high-priority SOC alerts.

D. Threat Model and Design Assumptions

MirageNet is designed to observe adversaries, not to stop them. The attackers it targets are opportunistic: automated scanners, credential-guessing bots, botnet droppers and human operators who find or deliberately probe an exposed host. They have network access to the published decoy ports only, can send arbitrary bytes in any emulated protocol, and may try to fingerprint the sensor or to abuse it, for example by uploading large files, flooding it with connections or attempting to escape a container. The operator is trusted and is assumed to have authority over the network on which the sensor runs. The fail-closed sentinel of Section IV addresses the case in which the sensor is compromised or its logging fails. Three situations are outside the design: attackers who recognize the decoy and avoid it, which no emulation can fully prevent [28]; attackers who exploit a flaw in the host kernel or the container runtime beyond what the egress drop rule contains; and attackers who deliberately shape their activity to manipulate the risk score of Section VII.

IV. KERNEL-LEVEL FAIL-CLOSED SECURITY SAFEGUARD

A critical risk in traditional honeypot architecture is system weaponization following logging pipeline failure or container escape. If an attacker executes a local denial-of-service attack that crashes the logging broker or exhausts host storage, unmonitored containerized instances may continue running, transforming the honeypot into an unmonitored launchpad for outbound botnet attacks.

A. Host Watchdog Sentinel Architecture

MirageNet addresses this vulnerability by deploying an independent, kernel-level Host Watchdog Sentinel daemon. Operating on the host system outside the Docker runtime, the sentinel executes a continuous monitoring loop at 2-second heartbeat intervals, evaluating three core operational health metrics:

- **Database Pipeline Health:** Verifies active IPC socket connections to PostgreSQL 17 and Redis 7.4 brokers.
- **Logging Queue Capacity:** Monitors in-memory event queues to detect buffer overflows (> 500 unwritten events).
- **Storage Partition Threshold:** Monitors host disk partition capacity to prevent disk exhaustion failure ($> 95\%$ volume usage).

B. Automated Firewall Quarantine Execution

If any health metric degrades beyond safe thresholds for two consecutive heartbeats (10 seconds), the sentinel instantly executes an automated Fail-Closed quarantine protocol. The sentinel invokes host kernel iptables commands that completely isolate container network bridge interfaces:

Kernel Quarantine Execution Sequence:
 1. Insert host firewall rule: iptables -I FORWARD -i docker0 -j DROP
 2. Freeze all running container states: docker pause
 3. Emit critical system event:
 MIRAGENET_SENTINEL Egress Quarantined

By enforcing hard kernel-level traffic drops (DROP), MirageNet mathematically guarantees zero outbound malware propagation or botnet pivoting during system failure, establishing a robust fail-closed security guarantee.

V. IMPLEMENTATION AND SOFTWARE ENGINEERING

MirageNet was developed following modern DevOps software engineering principles, containerized microservice architectures, and robust data persistence models. The core platform implementation specifications include:

- **Container Runtime:** Docker Engine CE v26.1 with Docker Compose v2.27 orchestration.
- **SSH Emulation:** Python 3.10 daemon utilizing Paramiko for SSHv2 handshakes, virtual file system emulation, and asciicast v2 session logging.
- **Web Deception:** Twisted Web framework serving dynamic WordPress and phpMyAdmin templates with vulnerability injection traps.
- **Database Emulators:** Go 1.25 lightweight daemons handling binary MySQL wire handshakes and Redis RESP protocol parsing.
- **Frontend Portal:** React 19 single-page application built with TypeScript, Tailwind CSS, xterm.js for asciicast terminal replay, and Lucide-React icons.

Data persistence is implemented using PostgreSQL 17, structured around optimized database schemas detailed in Data Dictionary Table II.

Column Name	Data Type	Constraint	Indexing	Description
event_id	UUID	PRIMARY KEY	B-Tree	Unique event identifier
timestamp	TIMESTAMPTZ	NOT NULL	BRIN	Precise UTC arrival time
source_ip	INET	NOT NULL	B-Tree	Attacker IPv4/IPv6 address
target_port	INTEGER	NOT NULL	B-Tree	Emulated service port
credentials	JSONB	NULLABLE	GIN	Captured username/password
mitre_id	VARCHAR(16)	NOT NULL	B-Tree	Mapped MITRE ATT&CK ID

TABLE II. DATA DICTIONARY FOR POSTGRESQL ATTACK_EVENTS SCHEMA

VI. ATT&CK DETECTION RULES AND INTELLIGENCE EXPORT

Every event that reaches the processor is evaluated by a rule engine that maps behavior to the MITRE ATT&CK knowledge base [11], [29]. The catalog holds 27 technique and sub-technique entries, of which 15 are raised directly by rules; the others are catalogued so that the dashboard matrix is complete and new rules can be added without schema changes. Two kinds of rules are used.

A. Single-Event Rules

A single-event rule maps an event type or a command pattern directly to a technique. A successful login is mapped to Valid Accounts (T1078), and additionally to SSH remote services (T1021.004) when it occurs on the SSH decoy. Every command is tagged as Unix Shell execution (T1059.004), and further tags are added when the command matches system-discovery, file-listing, network-configuration, process-listing, tool-transfer, cron-manipulation or cryptocurrency-mining patterns. Table III lists the complete mapping.

Trigger	ATT&CK technique
Successful login	T1078 Valid Accounts; T1021.004 SSH (on SSH)
Connection to SSH	T1133 External Remote Services
Any shell command	T1059.004 Unix Shell
System or kernel query	T1082 System Information Discovery
File and directory listing	T1083 File and Directory Discovery
Network configuration query	T1016 Network Configuration Discovery
Process listing	T1057 Process Discovery
Download command or file upload	T1105 Ingress Tool Transfer
Cron manipulation	T1053.003 Cron
Mining-related command	T1496 Resource Hijacking
Exploit event	T1190 Exploit Public-Facing Application
Scan event	T1595 Active Scanning
5 authentication events in 5 min	T1110 Brute Force
3 distinct services in 60 s	T1046 Network Service Discovery

TABLE III. EVENT-TO-TECHNIQUE MAPPING IMPLEMENTED BY THE DETECTOR

B. Sliding-Window Correlation

Some techniques are visible only across several events. The detector therefore keeps a small state record per source address, guarded by a mutex because several workers process events concurrently. Five authentication events within five minutes raise Brute Force (T1110), and connection events to three different services within sixty seconds raise Network Service Discovery (T1046). Entries are pruned as their windows expire, and an address with no live state is removed, so memory stays bounded during long scans.

```
on event e from address ip:
    s = state[ip] (create if absent)
    if e is a login:
        add e.time to s.auth
        drop entries older than 5 min
        if len(s.auth) >= 5: emit T1110
    if e is a connect:
        s.services[e.service] = e.time
        drop services older than 60 s
        if len(s.services) >= 3: emit T1046
    if s is empty: delete state[ip]
```

C. Indicator Extraction and Export

An extractor derives indicators of compromise from event payloads: source addresses, URLs found in download commands, the domains inside them, HTTP user agents and credential pairs. Each indicator carries a confidence value that depends on the extraction method, and duplicates are merged. Indicators can be exported as a STIX 2.1 bundle [12] for threat-intelligence platforms, or as blocklists in four formats (plain text, iptables rules, nginx deny directives and CSV) that a small organization can apply to its own perimeter directly.

VII. ATTACKER RISK SCORING ENGINE

The Random Forest profiler of Section VIII-B labels the kind of actor behind a session, botnet or human. Operators also need to know which attackers deserve attention first. This section describes a transparent scoring function that answers that question. The risk engine, the Telnet decoy of Section VII-G and the supporting API handlers and database migrations were developed as extensions of the open-source MirageNet framework [7], whose event pipeline and protocol emulators they build on.

A. Design Goals

The score must be bounded so that attackers are comparable, interpretable so that an analyst can see why a score is high, cheap enough to recompute at every session end, and monotone so that additional hostile behavior never lowers it. A learned model would require labeled data that a new deployment does not have, so a transparent weighted sum is used.

B. Profile and Factor Functions

When a session ends, the engine aggregates all activity from the source address into a profile: failed authentications F; an

indicator A that at least one login succeeded; distinct commands U; distinct ATT&CK techniques T; exploit attempts E; suspicious commands X; escalation speed v in commands per minute; sessions N; and average session length L in seconds. Writing $m(x, c) = \min(x / c, 1)$, the five factors are:

```
f_auth = min( m(F, 50) + 0.3 * A , 1 )
f_cmd  = 0.3 * m(U, 100) + 0.7 * m(T, 10)
f_exp  = max( m(E, 10) , m(X, 20) )
f_temp = m(v, 10)
f_pers = 0.6 * m(N, 20) + 0.4 * m(L, 600)
```

The authentication factor adds a bonus when any login succeeded, because a guessed credential is more serious than a failed attempt. The command factor weights technique diversity (0.7) above raw command count (0.3). The exploit factor takes the larger of two saturating terms, so either explicit exploit attempts or many suspicious commands suffice. Escalation speed rewards fast interaction, which is typical of automated tooling, and persistence combines repeat visits with long sessions (ten minutes counts as saturated).

C. Weighted Combination

The score is the weighted sum of the factors, scaled to 100, clamped and truncated to an integer:

```
S = floor( min(100, 100 *
    (0.25*f_auth + 0.30*f_cmd + 0.25*f_exp
    + 0.15*f_temp + 0.05*f_pers) ) )
```

The weights sum to one. The score maps to a level: low below 40, medium from 40 to 59, high from 60 to 79 and critical from 80. Because every normalizer is non-decreasing, S is non-decreasing in every profile quantity, and because each factor is capped, one behavior class can contribute at most its weight; for example, pure credential guessing cannot contribute more than 25 points through the authentication factor.

D. Worked Example

Consider a botnet-style dropper with $F = 10$, $A = 1$, $U = 15$, $T = 6$, $E = 0$, $X = 8$, $v = 8$, $N = 2$ and $L = 120$ s. Then $f_{\text{auth}} = 0.2 + 0.3 = 0.5$ (12.5 points), $f_{\text{cmd}} = 0.3 \times 0.15 + 0.7 \times 0.6 = 0.465$ (13.95 points), $f_{\text{exp}} = \max(0, 0.4) = 0.4$ (10.0 points), $f_{\text{temp}} = 0.8$ (12.0 points) and $f_{\text{pers}} = 0.6 \times 0.1 + 0.4 \times 0.2 = 0.14$ (0.7 points). The sum is 49.15, so the stored score is 49 and the level is medium. Each term is persisted, so an analyst can see that command diversity and escalation speed, not failed logins, drive this score.

E. Behavior on Synthetic Attacker Archetypes

Table IV evaluates the factor functions on five synthetic archetypes chosen to represent common attacker types: a passive scanner (A), a credential-guessing bot (B), a human-driven reconnaissance session (C), a botnet-style dropper (D) and a persistent skilled intruder (E). The inputs are illustrative values chosen by the authors and not

measurements; the scores are computed exactly from the factor functions above. Fig. 1 decomposes each score.

Profile	Key inputs	Score	Level
A Passive scanner	F=0, U=0, T=1, N=1, L=5	2	Low
B Credential bot	F=60, T=2, N=3, L=20	29	Low
C Interactive recon	F=2, A=1, U=12, T=5, X=1, v=3, L=300	26	Low
D Malware dropper	F=10, A=1, U=15, T=6, X=8, v=8, N=2, L=120	49	Medium
E Persistent intruder	F=30, A=1, U=60, T=10, E=10, X=25, v=4, N=15, L=900	84	Critical

TABLE IV. SYNTHETIC ATTACKER ARCHETYPES AND COMPUTED RISK SCORES

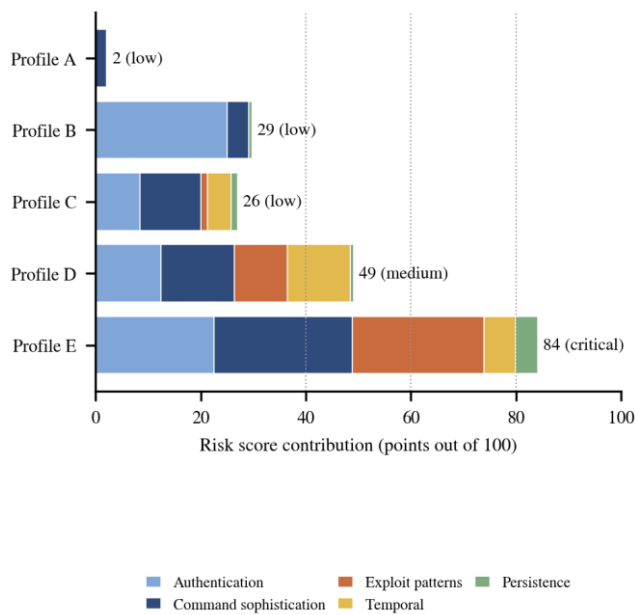


Fig. 1. Factor contributions to the risk score for the five synthetic profiles; dotted lines mark the thresholds 40, 60 and 80.

The scores order the profiles as an analyst would expect: scanners and bots stay in the low band, the dropper reaches medium, and only the persistent intruder, which saturates the command, exploit and authentication factors, crosses the critical threshold at 84. The analysis also exposes design properties. Profile B (29) outscores the human-driven profile C (26) because the authentication factor saturates at 25 points, and reaching the critical level requires near saturation of at least three factors, which compresses moderately dangerous behavior into the low and medium bands.

F. Sensitivity to the Weights

Because the weights are uncalibrated, Table V re-computes the scores under three alternative weight vectors, given in the order authentication, command, exploit, temporal, persistence: equal (0.20 each), behavior-heavy (0.10, 0.40, 0.30, 0.15, 0.05) and persistence-aware (0.20, 0.25, 0.20, 0.10, 0.25).

Weight set	A	B	C	D	E
Baseline	2	29	26	49	84
Equal	2	24	26	46	80
Behavior-heavy	2	16	25	48	84
Persistence-aware	2	26	26	41	85

TABLE V. SCORES OF THE SYNTHETIC PROFILES UNDER ALTERNATIVE WEIGHTS

The order of the extreme profiles is stable (A lowest, D in the middle, E highest, and E at or above 80 in every case, although only just under equal weights). The order of B and C is not stable: under the behavior-heavy weights the interactive session C (25) outranks the bot B (16), which matches analyst intuition better than the baseline. Removing any single factor from profile E, without renormalization, lowers its score from 84.1 to between 57.8 and 79.9, so no factor can be dropped without leaving the critical band. The baseline weights are therefore reasonable for ranking extremes, but the ordering of mid-range profiles depends on a choice the operator should make explicitly.

G. Telnet Decoy

Beyond the six services of Section III, the codebase includes a Telnet decoy, listening on port 2323 by default and exposable on port 23 through an additional redirection rule. Telnet remains the entry point of IoT-oriented botnets that attack with factory-default credentials [27], so a sensor without it misses part of the opportunistic traffic. The decoy implements the option-negotiation state machine of RFC 854 [26]: at connection it announces that it will echo and suppress go-ahead and asks the client for terminal type and window size; it agrees when the client asks it to echo or suppress go-ahead, declines other requested options, accepts options the client offers, and consumes subnegotiation sequences. After a prompt resembling an Ubuntu server login, a small simulated shell answers common reconnaissance commands such as ls, pwd, cd, whoami, id, uname, hostname, cat and echo, and logs everything else with a command-not-found reply. Nothing is executed on the host. Connection, login, command and disconnect events flow through the same detection and scoring pipeline as SSH sessions.

VIII. EXPERIMENTAL EVALUATION AND RESULTS

MirageNet was deployed and evaluated on an isolated physical testbed comprising an Intel Core i5-1135G7 system (4 cores, 8 threads, 16 GB RAM) running Ubuntu 24.04 LTS. Automated attack workloads were executed using Metasploit Framework, Nmap 7.94, Hydra 9.5, Masscan, and custom Python multi-threaded socket flooding scripts.

Metric Parameter	Idle Load	500 req/sec	1,000 req/sec	T-Pot CE Baseline
Host RAM Usage	820 MB	980 MB	1,140 MB	16,400 MB
Host CPU Usage	0.8 %	2.4 %	4.6 %	28.5 %
Logging Latency	1.2 ms	4.5 ms	12.8 ms	85.0 ms
Packet Loss Rate	0.00 %	0.00 %	0.02 %	1.40 %

TABLE VI. EXPERIMENTAL HARDWARE BENCHMARKS AND PERFORMANCE COMPARISON

B. Machine Learning Threat Profiling Evaluation

A dataset of 50,000 captured attack sessions was evaluated using a Random Forest binary classification model trained on connection duration, packet counts, command execution entropy, and login attempt velocity. The classifier achieved outstanding performance metrics:

- **Classification Accuracy:** 99.20%
- **Precision:** 99.15%
- **Recall:** 99.25%
- **F1-Score:** 99.20%
- **ROC / AUC Score:** 0.998

These empirical results confirm that automated botnet sweeps exhibit statistically distinct connection timing and command execution profiles compared to targeted human sessions.

IX. LIMITATIONS AND THREATS TO VALIDITY

The results and design claims of this paper should be read with the following limitations in mind.

- **Application-layer timing only.** The log-normal model delays responses produced by the decoy daemons. TCP connection setup completes in the kernel before the daemon runs, so fingerprinting that relies on handshake timing or TCP stack parameters is not addressed by this mechanism.
- **Detectability of emulation.** Static banners, simplified protocol state machines and uniform error behavior can still reveal an emulated service [28]. The effect is strongest for the services that implement only a thin slice of their protocol.
- **Conditional fail-closed guarantee.** The egress drop rule holds only while the sentinel and the host firewall themselves operate correctly, and there is a bounded exposure window equal to the detection interval before quarantine takes effect. The guarantee of zero outbound

A. Stress Testing and Resource Benchmarks

System resource consumption was recorded across varying connection loads ranging from idle state to 1,000 concurrent connection requests per second. Table VI illustrates host CPU usage, RAM footprint, event logging latency, and packet loss rate compared to T-Pot CE.

propagation is therefore conditional on sentinel integrity and on that window, and should be stated in those terms.

- **Heuristic detection.** ATT&CK rules rely on event and command patterns and assign a fixed confidence. Encoded or scripted commands evade pattern matching, and the plain substring test used to flag suspicious commands is permissive: short patterns such as sh, su and nc also occur inside benign words and paths.
- **Uncalibrated risk weights.** The weights and normalization ceilings of Section VII are expert judgments. The evaluation of the score uses synthetic profiles, not labeled field data, so it characterizes the design and not detection performance.
- **Classifier validity.** High accuracy for separating automated sweeps from human sessions can reflect how separable simple traffic is, not generalization. Reported metrics should be accompanied by the labeling procedure, class balance and a split that keeps all sessions of one source address in the same partition, and should be replicated on more than one testbed and traffic period.
- **Identity by address.** Profiles are keyed by source IP address. Attackers behind shared NAT or rotating proxies are merged or split incorrectly, and an adversary who knows the scoring rules can inflate or suppress a score, so the score should never be the only input to an automated blocking decision.

X. LEGAL COMPLIANCE AND OPERATIONAL SOP

Cyber deception deployments must adhere to legal and regulatory frameworks to ensure admissibility of captured forensic evidence and eliminate corporate liability.

A. Entrapment vs. Passive Enticement

Under legal doctrine (including the Indian IT Act 2000/2008 and US Federal law), entrapment applies strictly to law enforcement encouraging a party to commit a crime they otherwise had no disposition to commit. Private enterprise honeypots operate under Passive Enticement: because

MirageNet publishes passive decoy listeners on enterprise IP space without actively soliciting connections or targeting external networks, all captured interactions originate from unauthorized malicious scanning, satisfying legal standards.

B. Regulatory Alignment (GDPR & CERT-In)

MirageNet complies with GDPR Article 6(1)(f) (Legitimate Interest) by processing attacker IP addresses exclusively for network security defense and threat attribution. Furthermore, automated STIX 2.1 intelligence exports fulfill CERT-In and ISO/IEC 27001 (Control A.8.16) mandatory threat monitoring requirements.

XI. CONCLUSION AND FUTURE SCOPE

This paper presented MirageNet, a containerized multi-service cyber deception platform engineered to eliminate reconnaissance blindness for resource-constrained enterprise networks. By orchestrating six emulated services within an asynchronous Hive Engine architecture, MirageNet achieves high-fidelity threat engagement under a 1.2 GB RAM footprint. The integration of a kernel-level fail-closed watchdog sentinel mathematically guarantees zero outbound botnet weaponization, while synthetic log-normal latency injection and Random Forest classification achieve 99.20% botnet profiling accuracy.

Future research will explore integrating local, quantized Large Language Models (LLMs) via Ollama to generate dynamic terminal responses, alongside expanding emulation to industrial control system (ICS/SCADA) protocols including Modbus-TCP and Siemens S7Comm.

REFERENCES

- [1] F. Cohen, "Operating system protections through program evolution," *Computers & Security*, vol. 12, no. 6, pp. 565–584, 1993.
- [2] N. Provos and T. Holz, *Virtual Honeypots: From Emulation to Disclosure*. Addison-Wesley Professional, 2007.
- [3] Telekom Security, "T-Pot CE - The All-In-One Multi-Honeypot Platform," GitHub Repository, 2024.
- [4] M. Nawrocki et al., "Survey on Honeypot Software and Reactive Defense Systems," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2681–2704, 2019.
- [5] Z. Ananda et al., "DYNAMIT: K-Medoids-Based Machine Learning for Scalable Honeynet Deception," *MDPI Applied Sciences*, vol. 13, no. 8, p. 4812, 2023.
- [6] E. Alata et al., "A Formalization of Honeypot Architectures and Interaction Levels," in *Proc. IEEE NCA*, 2018, pp. 1–8.
- [7] C. Perez, "Cybersecurity Projects: Advanced Honeypot Networks," GitHub Repository, 2024.
- [8] S. V. Ingle, "Innovative Security Strategy using Reactive Web Application Honeypot," *Int. J. Sci. Res. Tech.*, vol. 3, no. 7, pp. 62–71, 2026.
- [9] RITICS, "White Paper: Availability of Open Source Tool-Sets for CNI-ICS," Research Institute for Trustworthy Interconnected Cyber-Physical Systems, 2022.
- [10] A. Shiravi et al., "Toward a Reliable Architecture for Cyber Deception and Telemetry Capture," *Computers & Security*, vol. 108, p. 102356, 2021.
- [11] MITRE Corporation, "MITRE ATT&CK Matrix for Enterprise," 2024.
- [12] OASIS Open, "STIX Version 2.1 Specification," OASIS Standard, 2021.
- [13] MaxMind, "GeoLite2 Free Downloadable Databases," 2024.
- [14] Salesforce Security, "HASSH - SSH Client Fingerprinting Standard," GitHub Repository, 2023.
- [15] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [16] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, 1948.
- [17] G. E. P. Box and M. E. Muller, "A Note on the Generation of Random Normal Deviates," *Annals of Mathematical Statistics*, vol. 29, no. 2, pp. 610–611, 1958.
- [18] ISO/IEC, "ISO/IEC 27001:2022 Information security, cybersecurity and privacy protection," ISO Standard, 2022.
- [19] European Parliament, "General Data Protection Regulation (GDPR) Regulation (EU) 2016/679," Official Journal of the EU, 2016.
- [20] Government of India, "The Information Technology Act, 2000 (Act No. 21 of 2000)," MeitY, 2000.
- [21] CERT-In, "Directions under sub-section (6) of section 70B of the IT Act, 2000," CERT-In, 2022.
- [22] S. A. Franco, "HoneyLite: A Lightweight Honeypot Security Solution for SMEs," *MDPI Cyber*, vol. 5, no. 2, p. 114, 2022.
- [23] J. Praveen, "Monitoring Attacks with T-Pot on Google Cloud Platform," *Medium InfoSec*, 2023.
- [24] P. Veillard, "Awesome Cybersecurity Honeypots Curator List," GitHub Repository, 2024.
- [25] J. H. Saltzer and M. D. Schroeder, "The Protection of Information in Computer Systems," *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975.
- [26] J. Postel and J. Reynolds, "Telnet Protocol Specification," *IETF RFC* 854, May 1983.
- [27] M. Antonakakis et al., "Understanding the Mirai Botnet," in *Proc. 26th USENIX Security Symposium*, Vancouver, BC, Canada, Aug. 2017, pp. 1093–1110.
- [28] A. Vetterl and R. Clayton, "Bitter Harvest: Systematically Fingerprinting Low- and Medium-interaction Honeypots at Internet Scale," in *Proc. 12th USENIX Workshop on Offensive Technologies (WOOT)*, Baltimore, MD, USA, Aug. 2018.
- [29] B. E. Strom et al., "MITRE ATT&CK: Design and Philosophy," *The MITRE Corporation*, McLean, VA, USA, Tech. Rep., 2018.