FLOW: EXHAUSTIVE TECHNICAL EDR SCRIPT

EDR ARCHITECTURE // COMPREHENSIVE SCHEMA-TO-CODE MAPPING // STACK JUSTIFICATION

Lead Architect: Abisin Raj | Technical Documentation v6.0 (Final Revision)

SLIDE 01: FLOW

TECHNICAL SCRIPT:

"**FLOW** is an offline-centric security guard for Linux. It connects network traffic directly to specific applications in real-time. Everything stays on the machine; no cloud needed."

SLIDE 02: WHAT IS FLOW?

TECHNICAL SCRIPT:

"Uses **Unified Telemetry** to map every data packet to the app that sent it. Built for **Data Sovereignty**—all processing happens locally. It creates a detailed audit trail by merging socket data with the system's `/proc` subsystem."

SLIDE 03: EXISTING SYSTEM (THE PROBLEM)

TECHNICAL SCRIPT:

"Current Linux monitoring is fragmented and manual. Tools like `netstat` and `tcpdump` don't talk to each other. Enterprise SIEMs are too resource-heavy for workstations. There is a **Correlation Gap**: seeing the traffic doesn't automatically reveal which app caused it.

Existing System Pros: No setup required; uses built-in tools.
Existing System Cons: No automated correlation; high manual effort; resource-heavy under load."

SLIDE 04: THE SOLUTION: A LIGHTWEIGHT EDR

TECHNICAL SCRIPT:

"Uses **Python** for logic and **PostgreSQL** for local data storage. Employs **POSIX Capabilities** to sniff traffic and manage firewalls without needing full root access.

Proposed System (Flow) Pros: Automatic app-to-traffic correlation; offline privacy; minimal CPU/RAM usage.
Proposed System (Flow) Cons: Requires initial installation; Linux-specific."

Stack Justification: PostgreSQL ensures security logs are persistent, searchable, and safe from power loss.

SLIDE 05: 1. USER (MODULE)

TECHNICAL SCRIPT:

"Hardware-accelerated dashboard showing real-time throughput and connections. Allows users to set **Smart Thresholds** for traffic volume and resolve threats by whitelisting or blocking apps with one click."

SLIDE 06: 2. LINUX KERNEL MITIGATION

TECHNICAL SCRIPT:

"Automatically links network traffic to app IDs using system files. Uses **nftables** to instantly block suspicious IPs or sandboxes problematic apps before they can cause damage."

CORE SYSTEM TOOLS:

`nftables` `psutil` `Scapy` `libcap`

SLIDE 07: HARDWARE & OS REQUIREMENTS

TECHNICAL SCRIPT:

"Runs on any modern Linux system with `/proc` support. Very low resource usage: 1GHz CPU and 256MB RAM minimum. Works on edge devices and old hardware without lagging the system."

SLIDE 08: SOFTWARE STACK & DEPENDENCIES

TECHNICAL SCRIPT:

"**Scapy** for packet reading, **TLSH** for fuzzy-matching malware signatures, and **psutil** for process mapping. **libcap** ensures minimum necessary permissions for security tasks."

SLIDE 09: DATA FLOW DIAGRAM (DFD) LEVEL 0

TECHNICAL SCRIPT:

"High-level flow: Raw network data and system states are ingested, processed through detection rules, and stored in the local vault. All data stays air-gapped on the machine."

SLIDE 10: DATA FLOW DIAGRAM (DFD) LEVEL 1 (USER INTERACTION)

TECHNICAL SCRIPT:

"Shows the connection between the UI and the data store. Actions like 'Block IP' update the database, which then triggers the firewall to cut the connection immediately."

SLIDE 11: DATA FLOW DIAGRAM (DFD) LEVEL 1.1 (SYSTEM DEEP-DIVE)

TECHNICAL SCRIPT:

"Direct view of the correlation engine. Simultaneously reads raw packets and polls `/proc` to identify which app owns each connection in real-time."

SLIDE 12: NETWORK & SYSTEM ARCHITECTURE DIAGRAM

TECHNICAL SCRIPT:

"Physical path: Internet -> Attacker -> ISP -> Router -> Local Host. Flow acts at the 'last hop,' inspecting the bridge between apps and the network where external firewalls can't see."

SLIDE 13: LOCAL TELEMETRY STORE - SCHEMA I

TECHNICAL SCRIPT:

"Database structure: Enriches every connection with app name, parent process, and GeoIP country code. Allows for fast forensic tracing."

`TABLE: core_connection (Socket Metadata)`

FIELD	LOGIC / INTERACTION	TECHNICAL JUSTIFICATION
pid / ppid	`get_proc_name_from_pid()`	Essential for detecting parent-child shell spawning cycles in EDR.
process_name	`/proc/{pid}/exe`	Absolute path tracking allows for secure binary whitelisting.
src_country	`lookup_ip() (GeoIP2)`	Local MaxMind resolution ensures 100% offline geospatial forensics.

`TABLE: core_alert (Audit Trail)`

FIELD	LOGIC / INTERACTION	TECHNICAL JUSTIFICATION
alert_type	`classify_scan()`	Categorizes threats (e.g., 'Fast Scan' vs 'Stealth Scan') via temporal analysis.
connection_id	`ForeignKey(Connection)`	Enables analyst pivoting from high-level alerts to raw network packets.

SLIDE 14: LOCAL TELEMETRY STORE - SCHEMA II

TECHNICAL SCRIPT:

"Quarantine and block-list management. Uses **fuzzy matching** to find polymorphic malware variants. Manages temporary IP blocks with automatic expiration."

`TABLE: core_quarantinedfile (Sandbox)`

FIELD	LOGIC / INTERACTION	TECHNICAL JUSTIFICATION
tlsh	`calculate_hashes()`	**Fuzzy Hash**: Superior to SHA256 for catching polymorphic variants.
match_distance	`Fuzzy Match logic`	Smaller distances indicate higher confidence in variant detection.

`TABLE: core_blockedip & core_watchedfolder`

FIELD	LOGIC / INTERACTION	TECHNICAL JUSTIFICATION
expires_at	`Dynamic Decay logic`	Prevents permanent IP blocking, allowing for auto-remediation.
path	`FolderWatcher events`	Real-time filesystem monitoring for dropper detection (e.g., in /tmp/).
auto_quarantine	`on_created()`	Automated immediate file isolation based on signature matches.

SLIDE 15: TESTING & VERIFICATION STRATEGY

TECHNICAL SCRIPT:

"Verification uses **Pytest** for logic, **Ruff** and **Mypy** for code quality, and **Bandit** for security scans. Entropy tests ensure zero data leakage from the local host."

EDR VERIFICATION SUITE:

`Pytest (Unit)` `Ruff (Lint)` `Mypy (Types)` `Bandit (Security)`

Architecture Note: PyQt6 ensures a responsive UI, while **libcap** enforces the principle of least privilege.

REFERENCE: KEY TECHNICAL SYNTAX

TECHNICAL SCRIPT:

"Direct code patterns used in Flow's implementation:"

1. Process-to-Socket Mapping (psutil)

```
            for proc in psutil.process_iter(['pid', 'name']):
            # Find established network connections for this PID
            connections = proc.connections(kind='inet')
            for conn in connections:
            if conn.status == 'ESTABLISHED':
            remote_ip = conn.raddr.ip
```

2. Firewall Mitigation (nftables)

```
            # Blocking an IPv4 address immediately
            subprocess.run(["nft", "add", "element", "inet", "flow_table",
"flow_blocked_ipv4",
            "{ 192.168.1.100 timeout 3600s }"])
```

3. Fuzzy Malware Hashing (TLSH)

```
            # Compute TLSH and check distance to known malware
            t_hash = tlsh.hash(file_data)
            dist = tlsh.diff(t_hash, known_malware_hash)
            if dist < 50: # High similarity threshold quarantine_file(path)
```

4. Secure Privilege Separation (Unix Socket)

```
# Core app talks to Root Helper via privileged socket
sock = socket.socket(socket.AF_UNIX, socket.SOCK_STREAM)
sock.connect("/run/flow/firewall.sock")
sock.sendall(f"BLOCK_IP {ip} {reason}".encode())
```

REFERENCE: DATABASE TABLE DESIGN

TECHNICAL SCRIPT:

"Exhaustive schema design for the FLOW telemetry store:"

`Table: core_connection (Network Events)`

COLUMN NAME	DATA TYPE	CONSTRAINT	WHAT IT'S FOR
id	AutoField	PK, Unique	Primary key for the connection record.
src_ip / dst_ip	GenericIPAddress	NotNull, Indexed	Unpacked IPv4/IPv6 addresses.
src_port / dst_port	PositiveInteger	NotNull	TCP/UDP port numbers.
pid	Integer	NotNull, Indexed	Owner process ID at the time of connection.
process_name	CharField(200)	NotNull	Name of the parent executable.

`Table: core_alert (Security Incidents)`

COLUMN NAME	DATA TYPE	CONSTRAINT	WHAT IT'S FOR
alert_type	CharField(100)	NotNull, Indexed	Category (e.g., 'Reverse Shell', 'Scan').
severity	CharField(20)	Default: 'medium'	Priority level for analyst review.
connection_id	ForeignKey	FK (set_null)	Link to the raw connection event.
message	TextField	NotNull	Full technical details of the detection.

Table: core_quarantinedfile (Malware Repository)

COLUMN NAME	DATA TYPE	CONSTRAINT	WHAT IT'S FOR
filename	CharField(255)	NotNull	Original name of the suspicious file.
sha256	CharField(64)	NotNull, Indexed	Unique file hash for exact matching.
tlsh	CharField(128)	NotNull	Fuzzy hash for similarity analysis.
match_distance	Integer	Nullable	Confidence score of the fuzzy match.

Table: core_blockedip (Firewall State)

COLUMN NAME	DATA TYPE	CONSTRAINT	WHAT IT'S FOR
ip	CharField(45)	Unique, NotNull	Target IP address to drop.
expires_at	DateTimeField	Nullable	Expiration time for auto-unblocking.
reason	TextField	NotNull	Audit note on why the IP was blocked.

SLIDE 16: QUESTIONS?

TECHNICAL SCRIPT:

"Technical summary: Private, lightweight Linux EDR. Focus: Kernel integration and automated response. Open for questions on architecture or design."