Documentation & Interpretation of the Idea

1. Initial Reconnaissance

Attacker Goals

Trying to map organization to find weakest link.

- **Infrastructure Mapping** : Identifying active hosts (computers/servers) and their IP addresses **to understand the network topology**.
- **Service Enumeration** : Discovering open ports (e.g., 80, 443, 3389) and the specific services running on them (e.g., "Apache 2.4.49" or "Windows IIS") **to find exploitable versions**.
- **Vulnerability Identification** : Finding known weaknesses in the discovered services (e.g., unpatched servers).
- **Employee Intelligence** : Gathering details about employees (names, emails, habits) via social engineering or open sources to prepare for Spear Phishing.
- **Security Posture Discovery** : Identifying what firewalls, anti-virus, or routers are in place.

What to Detect ?

Detection relies heavily on **Network Traffic Analysis** (using Packetbeat) and **Web Server Logs.**

A. IP Scanning (Ping Sweeps)

- **What it is:** The attacker sends ICMP (Ping) packets to a range of IPs (e.g., `192.168.1.1` to `192.168.1.255`) to see which ones answer.
- **How to detect in ELK:** Monitor **Packetbeat** (ICMP traffic) flows.
- **The Logic:**

> **IF** `source.ip` sends ICMP Echo Requests to > 10 distinct `destination.ip` addresses within 1 minute **THEN** Trigger Alert: "Potential Ping Sweep".

B. Port Scanning (Vertical Scan)

- **What it is:** The attacker finds a live server and checks *every* port (1-65535) to see what is open.
- **How to detect in ELK:** Monitor **Packetbeat** or **Firewall Logs** for multiple connection attempts (SYN packets).

- **The Logic:**

> **IF** `source.ip` initiates connections (TCP SYN) to > 15 distinct `destination.port` numbers on the SAME `destination.ip` within 30 seconds **THEN** Trigger Alert: "Vertical Port Scan Detected".

C. Vulnerability Scanning (Web Probing)

- **What it is:** The attacker uses tools like **Nessus**, **Nikto**, or **Burp Suite** to automatically check your web servers for thousands of known bugs.
- **How to detect in ELK:** Monitor **HTTP Logs** (via Packetbeat or Nginx/Apache Filebeat modules).
- **The Logic (Signature Based):**

> **IF** `http.user_agent` contains "Nessus", "Nmap", "Nikto", or "Sqlmap" **THEN** Trigger Alert: "Vulnerability Scanner Detected".

> **OR (Behavioral): IF** `source.ip` generates > 50 `404 Not Found` errors within 10 seconds (implies they are guessing file paths that don't exist) **THEN** Trigger Alert: "Web Directory Brute Force".

D. Threat Intelligence Correlation / Passive Reconnaissance

- **What it is:** Sometimes reconnaissance looks like normal traffic (e.g., browsing your website).
- **How to detect in ELK:** Integrate **MISP** (Malware Information Sharing Platform).
- **The Logic:**

> **IF** `source.ip` matches an IP in the MISP database (Known Threat Actor) **THEN** Trigger Alert: "Reconnaissance from Known Malicious Actor".

Initial Compromise / Weaponization

Attacker Goals

The attacker needs to execute malicious code on a victim's machine to establish a beachhead.

- **Code Execution:** Getting the victim's computer to run unauthorized code (malware, script, or exploit).
- **Malware Delivery:** successfully transferring a payload (Trojan, Backdoor, or Dropper) onto the target system.

- **Bypassing Perimeter Defenses:** Using trusted channels (like Email or Web Traffic) to slip past firewalls.
- **Exploitation:** Leveraging vulnerabilities in browsers (Watering Hole attacks) or applications (Office/PDF exploits) to gain control.

Spear Phishing (Malicious Attachments)

- **The Scenario:** A victim receives an email (masquerading as DHL, HR, or Finance) with an attachment (Word, Excel, PDF) .
- **Host Detection (Sysmon):**
  - **Process Chain Analysis:** Normal users open Word to type. They do *not* open Word to run system commands. If `winword.exe` spawns `powershell.exe` or `cmd.exe` , it is almost certainly a malicious Macro.
  - **File Creation:** Detect when an Office application writes an executable file (`.exe` , `.scr` , `.js`) to disk (Sysmon Event ID 11).
- **Network Detection (Packetbeat):**
  - **Suspicious Downloads:** Detecting downloads from non-business domains or encrypted zip files.

System Architecture (Host + Network)

To detect this, you need a "pincer maneuver"—catching the payload arrival on the network and its execution on the host.

- **Network Sensor (Packetbeat/Suricata):**
  - **Role:** It sees the delivery vector (the email attachment or the file download) before it executes.
  - **Data:** HTTP/HTTPS flows, DNS queries, TLS certificate info.
- **Host Sensor (Sysmon + Auditd):**
  - **Role:** It sees what happens *after* the user clicks. It monitors process creation, file creation, and command-line arguments.
  - **Data:** Process ancestry (`ParentImage` vs. `Image`), file hashes, network connections from non-browser processes.

Scenario A: Spear Phishing (Malicious Attachment)

- **Host Detection (Sysmon):**
  - **What:** Normal users open Word to read. They do *not* use Word to launch scripts.
- **Logic (Process Ancestry):**

> **IF** `ParentImage` is `winword.exe` OR `excel.exe` OR `outlook.exe` **AND** `Image` (Child) is `powershell.exe` OR `cmd.exe` OR `wscript.exe` **THEN** Trigger Alert: **"Suspicious Office Process Hierarchy"**.

- **Advanced Logic (File Creation):**

> **IF** `Image` is `winword.exe` **AND** `TargetFilename` ends with `.exe` , `.scr` , `.js` , or `.ps1` **THEN** Trigger Alert: **"Office Application Dropping Executable"**.

- **Network Detection:**
  - **Logic (MIME Type Mismatch):** Detect "Distinguished EXE files" where the file extension says `.pdf` or `.jpg` , but the actual content (Magic Bytes) is an Executable (`MZ` header).

System Architecture

While this is primarily a **Host-Level** detection game, the Network layer provides corroborating evidence.

- **Host Sensor (Sysmon + Winlogbeat):**
  - **Role:** The "Change Monitor." It watches for changes to the operating system's "Auto-Start Extensibility Points" (ASEPs).
  - **Data:** Process Creation (Event 1), Registry Events (Event 12/13), File Creation (Event 11).
- **Network Sensor (Packetbeat):**
  - **Role:** The "Download Monitor."
  - **Data:** Often, the persistence mechanism ("The Task") triggers a network call to a C2 server to download an update or payload. You detect the *traffic* generated by the persistence mechanism.

System Architecture

This phase is almost entirely **Host-Centric**. The network layer provides little visibility until the credentials are *used* (Phase 6).

- **Host Sensor (Sysmon):**
  - **Role:** It watches specifically for processes trying to touch the sensitive parts of the OS (like the Local Security Authority Subsystem Service - `lsass.exe`).

    - **Critical Events:**
        - **Event ID 1:** Process Creation (Catching the tool execution).
        - **Event ID 10:** Process Access (Catching the memory read attempt).
- **Threat Intel Integration (VirusTotal):**
    - **Role:** Validating the tools. Attackers often rename `mimikatz.exe` to `mimidog.exe` or `update.exe` . Checking the file hash against VirusTotal bypasses this renaming trick.

Internal Reconnaissance

Attacker Goals

They land on a random workstation (e.g., "HR-PC-05") and need to find the path to the critical data (e.g., "Database-Server-01"). This phase is about **Situational Awareness** .

- **Context Discovery:** "Where am I?" (Host info, OS version, Patches).
- **Privilege Discovery:** "Who am I?" (Current user rights, Groups).
- **Network Discovery:** "Where can I go?" (Routes, ARP table, DNS cache).
- **Account Discovery:** "Who else is here?" (Domain Admins, Logged on users).

Living off the Land (LOLBins) Attackers rarely bring custom scanning tools here because antivirus would catch them. Instead, they use **built-in Windows commands** (`net.exe` , `whoami.exe` , `systeminfo.exe`) because these are trusted and "clean."

System Architecture

This phase is 90% **Host-Centric**. While some reconnaissance generates network traffic (LDAP queries to the Domain Controller), the *execution* happens on the endpoint.

- **Host Sensor (Sysmon):**
    - **Role:** It captures every command typed into `cmd.exe` or `powershell.exe` .
    - **Critical Events: Event ID 1** (Process Create). This is the gold mine. It shows the `Image` (tool used) and `CommandLine` (what it asked for).
- **Context Layer (Logstash):**
    - **Role:** Differentiating "Admins" from "Attackers." Sysmon just sees `whoami.exe` running. Logstash needs to know *who* ran it to decide if it's suspicious.

Scenario A: The "Scripted" Burst (Noise)

- **The Tactic:** The attacker runs a "recon script" (batch file) that fires off 10 commands in 2 seconds to dump everything.

- **Commands:** `ipconfig /all` , `net user /domain` , `net group "Domain Admins" /domain` , `systeminfo` , `tasklist` , `netstat -ano` .
- **Detection Logic (Sequence / Threshold):**
  - **Concept:** Normal users don't run 5 different networking commands in 10 seconds.
  - **Rule:**

> **IF** `User` executes (> 5 unique processes) from list [`net.exe` , `ipconfig.exe` , `whoami.exe` , `systeminfo.exe` , `nltest.exe` , `route.exe`] **WITHIN** 1 Minute **THEN** Trigger Alert: **"Rapid Internal Reconnaissance Activity"**.

Scenario B: The "Slow" Manual Recon (Stealth)

- **The Tactic:** The attacker manually types one command, reads the output, thinks for 5 minutes, then types another.
- **Detection Logic (Context / Anomaly):**
  - **Concept:** HR or Finance employees *never* need to check `arp -a` or `route print` .
  - **Rule:**

> **IF** `Image` is `whoami.exe` OR `net.exe` **AND** `User` is NOT in "IT_Admins_Group" (Watchlist) **THEN** Trigger Alert: **"Suspicious Discovery Command by Non-Admin"**.

Scenario C: Active Directory Enumeration (LDAP)

- **The Tactic:** Using `net view` or PowerShell to list all computers in the domain.
- **Detection Logic (Network/Host Hybrid):**
  - **Host:** Detect `net.exe view` or `dsquery` .
  - **Network:** Detect a spike in **LDAP** traffic (Port 389) from a workstation to the Domain Controller.

System Architecture

Detection here is difficult because it looks like "System Administration." The key is **Correlation**—linking the *Source* behavior to the *Destination* event.

- **Host Sensor (Sysmon - Source & Destination):**
  - **Source:** Captures the execution of the tool (e.g., `psexec.exe` running on Client A).
  - **Destination:** Captures the *result* (e.g., `PSEXESVC` service installation or `Logon Type 3` on Server B).
- **Network Sensor (Packetbeat):**
  - **Role:** The "Traffic Cop." It monitors **SMB (Port 445)** and **RDP (Port 3389)**.

    - **Insight:** It sees the *volume* and *timing*. A user connecting to 50 machines in 1 minute via SMB is distinct network behavior (Graph Anomaly).

Scenario A: Pass-the-Hash (PtH) / Credential Reuse

- **The Tactic:** Using a stolen NTLM hash to authenticate to another machine via SMB.
- **Specific Log:** Windows Security **Event ID 4624** (Logon) on the **Destination** machine.
- **Detection Logic (Rule-Based):**
    - **Concept:** In a healthy Active Directory environment, workstations usually use **Kerberos**. If a workstation authenticates using **NTLM** (Logon Type 3), it is suspicious, especially if associated with admin accounts.
    - **Rule:**

> **IF** `LogonType` is `3` (Network) **AND** `AuthenticationPackage` is `NTLM` **AND** `LogonProcess` is `seclogon` OR `NtLmSsp` **AND** User is NOT "ANONYMOUS LOGON" **THEN** Trigger Alert: **"Potential Pass-the-Hash (NTLM over Network)"** .

1. Anomalous Login: Bipartite Anomaly (Unsupervised ML)

The Concept: This method moves beyond looking at a single log line. Instead, it models the entire network's login behavior as a **Bipartite Graph**.

- **Bipartite** means the graph has two distinct sets of nodes: **Users** on one side and **Computers** on the other.
- **Edges (Lines)** represent a successful login.

How it Works (The Logic):

1. **Mapping:** The system ingests authentication logs and draws lines between Users and the Machines they access.
2. **Baselining:** It learns the "normal structure." For example, "Marketing Users" (Group A) only connect to "Marketing Workstations" (Group B).
3. **Scoring (The Anomaly):** It calculates a **Bipartite Anomaly Score**.
    - If a User from Group A suddenly connects to a Machine in Group C (e.g., a Database Server), the graph structure changes.
    - The algorithm flags this edge because it mathematically deviates from the established user-machine relationship clusters.

System Architecture

While C2 is a network phenomenon, correlating it with Host data eliminates false positives (like a weather widget updating every hour).

- **Network Sensor (Packetbeat):**
  - **Role:** Captures the **Interval** and **Volume**.
  - **Data:** HTTP flows, DNS queries.
- **Host Sensor (Sysmon):**
  - **Role:** Captures the **Process** responsible.
  - **Data: Event ID 3** (Network Connection).
  - **Critical Check:** "Is `chrome.exe` making this connection (Normal)? Or is `powershell.exe` making this connection to the internet (Suspicious)?"

Method A: Periodicity Analysis (Statistical / "Time Transforms")

- **The Concept:** Human behavior is random. We don't click a link *exactly* every 300 seconds. Automated malware *does*. Even with "jitter" (randomness added), the mathematical underlying frequency remains detectable.
- **The Model: Time Transforms** or **Fast Fourier Transform (FFT)**.
- **How it works:**
  1. Collect timestamps of all connections from `Host A` to `Domain B` .
  2. Calculate the time difference (Δt) between consecutive packets.
  3. Analyze the variance. If the connections show a strong **Periodicity** (regular pattern), it is likely a C2 beacon .

Method B: Graph-Based Detection (Unsupervised ML)

- **The Concept:** C2 domains have a different "social network" than normal domains (like Google or Facebook).
- **The Model: Bipartite Graphs** (e.g., the "Segugio" system).
- **How it works:**
  1. Build a graph linking **Hosts** to **Domains**.
  2. **Normal Domains:** Connected to *many* hosts (dense).
  3. **C2 Domains:** Connected to only a *few* specific infected hosts (sparse) but with high frequency.
  4. The ML algorithm clusters these nodes. If a domain is "unknown" (new) and has a "sparse but active" graph structure, it is flagged as malware-controlled .

Property **+** **+** **+**	**Normal Domain (e.g., Google)** **+**	**C2 Domain (e.g., h4x0r.com)** **+**
Fan-out (Degree) +	**High** (1000s of unique internal IPs)	**Low** (1 to 5 unique internal IPs)
Activity (Weight) +	**Medium-High** (Regular use, but dispersed)	**Very High** (High query rate from a single host: **Beaconing**)

The ML model finds the smallest, most isolated cluster and flags all domains within it as potential **C2 channels**, proving the hypothesis that C2 domains inhabit a different "social network" within the graph.

`run_graph_method()` : Implements your **Advanced Hypothesis**. It uses **NetworkX** to build the Bipartite Graph for each dataset and applies the **Segugio Logic**:

- **Filter 1 (Isolation/Sparsity):** `Degree < 2` (Only 1 or 2 internal hosts are talking to it).
- **Filter 2 (Activity/Volume):** `Weight > 100` (The talking is frequent).
- This method is designed to catch the C2 channel by its anti-social structure, not just its total volume, hence providing the **accuracy advantage**.

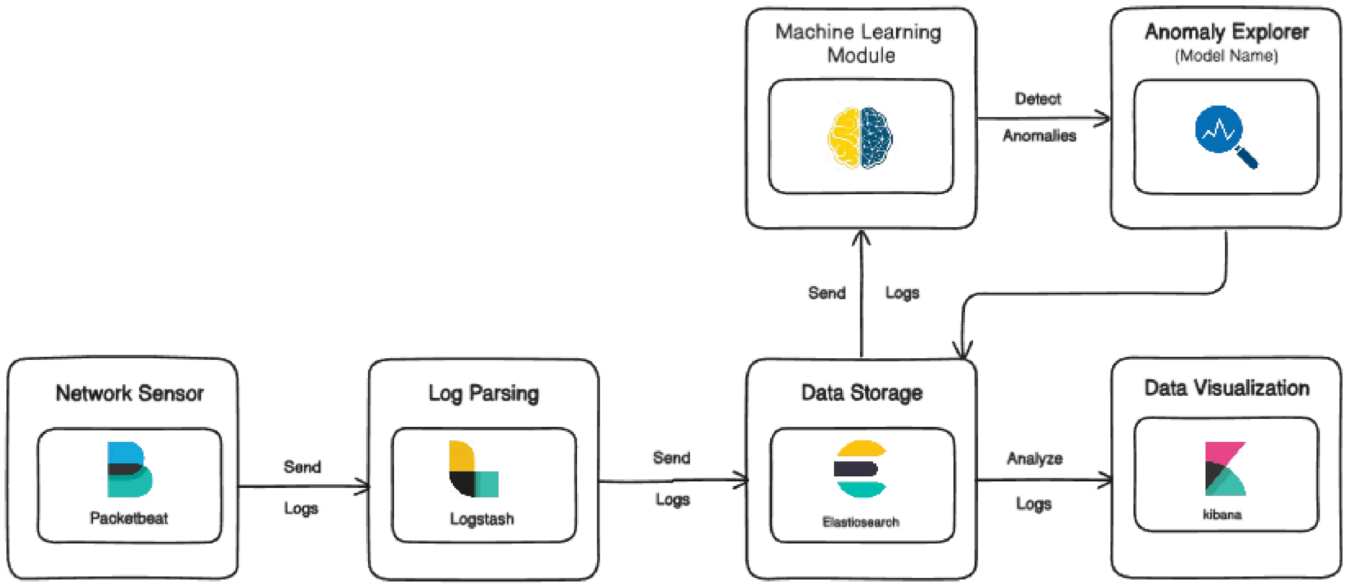
Scenario A: DNS Exfiltration (Tunneling)

- **The Tactic:** The attacker breaks a file (e.g., `secret.pdf`) into binary chunks, encodes them in Base64, and sends them as subdomains to a server they control.
    - *Example Query:* `[base64_chunk_1].attacker.com`
- **Detection Logic (Entropy Analysis):**
    - **Concept:** Normal domains (`google.com` , `finance.yahoo.com`) are readable English. Encoded data (`Ls4Dd58nRQJh...`) looks like random noise.
    - **The Math:** We use **Shannon Entropy**. High randomness = High Information Content.
    - **ML Job:** Detect when the **Information Content (Entropy)** of a subdomain is significantly higher than the average for that domain .

Scenario B: HTTP Exfiltration (Upload Volume)

- **The Tactic:** The attacker POSTs data to a web server.
- **Detection Logic (Ratio/Volume Analysis):**
  - **Concept:** In normal web browsing, clients **Download** (GET) far more than they **Upload** (POST). If a client uploads 500MB to an external IP, it is an anomaly.
  - **ML Job:** Detect "High Sum" of `bytes_out` (or `bytes_in` from the server's perspective) relative to the specific destination host .

Machine Learning Part



(Exfiltration): Deep Reinforcement Learning (DRL)

We previously discussed Entropy (Static ML). **Deep Reinforcement Learning** offers a *dynamic* defense against attackers who "learn" to evade your static thresholds.

- **Goal:** Catch "Low and Slow" exfiltration that adapts to your defense.
- **Detection Focus:** Network Level.
- **What to Detect:**
  - Attackers who dynamically change their packet size or timing to stay *just below* your "High Sum" alert threshold.
- **How it Works (The Logic):**
  - **The Agent:** An ML Agent observes the network environment state (traffic volume, frequency).
  - **The Reward:** The agent gets a "reward" for correctly identifying malicious traffic and a "penalty" for false positives or missed attacks.

    - **Dynamic Policy:** Unlike a fixed rule (`if bytes > 5MB`), the DRL agent learns a **Policy**. It might learn: *"If traffic volume drops but frequency increases, this is an evasion attempt."*
    - **Performance:** Research shows DRL approaches (like APT-DRL) outperform static Neural Networks in learning speed and accuracy for uncertain traffic flows .

All Phases (Model Trust): Explainable AI (XAI) - SHAP & LIME

A major issue with advanced ML (like the Autoencoders or LSTMs we discussed) is that they are "Black Boxes." An analyst might see "Alert: Anomaly Score 99%" but not know *why*. **XAI** solves this.

- **Goal:** Provide transparency and trust. Tell the SOC analyst *exactly* which feature triggered the alert.
- **Detection Focus:** Post-Detection Analysis (The "Why").
- **What to Detect:** The specific attribute driving the anomaly (e.g., "Was it the Time of Day? Or the Packet Size?").
- **How it Works (The Logic):**
    - **SHAP (Shapley Additive Explanations):** It assigns a "contribution value" to every feature.
        - *Scenario:* The ML model flags a `PowerShell` execution.
        - *SHAP Output:* "This is anomalous because: `ParentImage = WinWord` (+50% contribution), `Time = 3 AM` (+20% contribution), `User = HR_Intern` (+30% contribution)."
    - **LIME (Local Interpretable Model-Agnostic Explanations):** It tweaks the input slightly to see if the prediction changes. It builds a simple, explainable model around that specific single prediction to explain it .