# **Cybersecurity Swarm: Master Blueprint**

This document provides a comprehensive, multi-layered breakdown of the cybersecurity agent swarm architecture. It details each component from a high-level overview down to its internal processes, inter-agent communications, and the specific technologies it employs.

## **1. SWARM CORE: Command & Control**

The core of the swarm consists of the Orchestrator, which acts as the central intelligence, and the Shared Knowledge Base, which serves as the collective memory.

### **1.1 Core Components Overview**



| **Component** | **Role & Function** | **Key Interactions** |
| --- | --- | --- |
| **Orchestrator (C2)** | The "Logic Pipe" and central brain of the swarm. | - **Assigns Tasks:** Dispatches tasks to appropriate Red and Blue Team agents. - **Receives Events:** Ingests all reports and events from agents. - **Makes Decisions:** Correlates events and triggers responsive or adaptive actions in other agents. |
| **Shared Knowledge Base (KB)** | Central repository for all operational data. Acts as the swarm's memory, enabling contextual awareness and historical analysis. | - **Receives Writes:** All agents report their findings (recon data, vulnerabilities, logs, defense actions) to the KB. - **Allows Reads:** All agents can query the KB for context to inform their tasks (e.g., a Red agent checks for defenses before acting). |

### **1.2 Orchestrator (C2) - Deep Dive**

* **Internal Process Flow:**
  1. **Agent Registration:** New agents connect and announce their capabilities.
  2. **Task Queue Management:** Maintains a queue of pending tasks.
  3. **Message Ingestion:** Continuously receives standardized Message objects from all agents.
  4. **Message Processing ("Logic Pipe"):**
     + Parses incoming messages (EventType, Payload).
     + Updates the Shared Knowledge Base with reported data.
     + Executes rules based on the EventType (e.g., VULNERABILITY\_FOUND from Red triggers a Blue response task).
  5. **Task Delegation:** Assigns tasks to appropriate agents based on their type and availability.
* **Typical Technologies:**
  + **Core Framework:** Python (asyncio, threading).
  + **Messaging:** Internal queues (deque), or scalable brokers like Kafka, RabbitMQ.
  + **KB Interface:** Database drivers (psycopg2 for PostgreSQL, pymongo for MongoDB).
  + **Decision Logic:** Rule engines (pyDatalog) or Machine Learning models (Reinforcement Learning).

## **2. RED TEAM: Offensive Operations**

The Red Team's mission is to simulate attacks, identify vulnerabilities, and test the target's defenses.

### **2.1 Red Team Overview**

| **Agent** | **Key Tasks (Functions)** | **Output / Event Reported to Orchestrator** |
| --- | --- | --- |
| **Discovery Agent** | network\_scan, port\_scan, service\_enum | **RECON\_DATA** (Live hosts, open ports, services) |
| **OSINT Agent** | osint\_collection, domain\_analysis, employee\_profiling | **RECON\_DATA** (Subdomains, emails, public info) |
| **Vulnerability Scanner** | vuln\_scan, config\_audit, webapp\_scan | **VULNERABILITY\_FOUND** (CVEs, misconfigurations) |
| **Exploitation Agent** | execute\_exploit, deliver\_payload, post\_exploit\_actions | **ACCESS\_GAINED** or **DATA\_EXFILTRATED** |
| **Strategy Adaptation Agent** | adapt\_attack\_strategy, re\_evaluate\_target | **ATTACK\_ADAPTATION** (Report on new strategy) |

### **2.2 Red Team Agent Details**

*Details in Appendix A.*

## **3. BLUE TEAM: Defensive Operations**

The Blue Team's mission is to detect, analyze, and respond to the Red Team's activities.

### **3.1 Blue Team Overview**

| **Agent** | **Key Tasks (Functions)** | **Output / Event Reported to Orchestrator** |
| --- | --- | --- |
| **Network Monitor Agent** | traffic\_analysis, flow\_monitoring, intrusion\_detection | **INTRUSION\_DETECTED** (Suspicious network activity) |
| **Log Analysis Agent** | log\_collection, log\_parsing, log\_anomaly\_detection | **INTRUSION\_DETECTED** (Anomalous log events) |
| **Threat Hunter Agent** | proactive\_monitor, behavioral\_analysis, ioc\_scan | **INTRUSION\_DETECTED** or **THREAT\_HUNT\_RESULT** |
| **Patch Management Agent** | remediate\_vuln, enforce\_config | **DEFENSE\_ACTION** (Report on patch/config applied) |
| **Containment Agent** | network\_isolate, process\_terminate, block\_ip\_domain | **DEFENSE\_ACTION** (Report on containment action) |

### **3.2 Blue Team Agent Details**

*Details in Appendix A.*

## **4. The Logic Pipe: Key Interaction Flows**

The "Logic Pipe" within the Orchestrator facilitates the adaptive nature of the swarm.

#### Flow 1: Red Discovers -> Blue Reacts

1. A **Vulnerability Scanner** reports VULNERABILITY\_FOUND.
2. The **Orchestrator** logs this in the KB.
3. The **Orchestrator** triggers the **Patch Management Agent** with a remediate\_vuln task.

#### Flow 2: Blue Detects -> Red Adapts

1. A **Network Monitor** or **Log Analysis Agent** reports INTRUSION\_DETECTED.
2. The **Orchestrator** logs this in the KB.
3. The **Orchestrator** triggers the **Strategy Adaptation Agent** with an adapt\_attack\_strategy task.

#### Flow 3: Blue Defends -> Red Re-evaluates

1. A **Patch Management** or **Containment Agent** reports DEFENSE\_ACTION.
2. The **Orchestrator** logs this in the KB.
3. The **Orchestrator** triggers the **Strategy Adaptation Agent** with a re\_evaluate\_target task.

## **Appendix A: Adversarial Risk Analysis & Hardening**

### Introduction

This analysis treats the swarm architecture as a potential target. The objective is to identify and categorize risks at every level, from the high-level orchestration down to the specific tools used by individual agents. By exposing these potential failure points, we can build in resilience and security from the ground up.

**Risk Categorization:**

* **Critical:** Compromise could lead to complete system failure, catastrophic misdirection of the swarm, or use of the swarm as a weapon against unintended targets.
* **High:** Compromise could lead to significant mission impairment, data loss, or evasion of core defensive functions.
* **Medium:** Compromise could degrade performance, produce inaccurate results, or require significant manual intervention.
* **Low:** Compromise has minimal impact on core functions but could be a stepping stone for more significant attacks.

### I. Core Components Risk Analysis

#### 1. Orchestrator (C2)

* **Overall Risk Level:** **Critical**
* **Rationale:** As the central brain, its compromise means total loss of control. An attacker could turn the swarm on its owners, shut it down, or feed it false intelligence.
* **Micro-Detailed Task Analysis:**
  + **Task: Agent Registration & Authentication:**
    - **Tools:** mTLS with a private Certificate Authority (CA).
    - **Risks:**
      * **CA Compromise (Critical):** If the CA private key is stolen, an attacker can generate valid certificates and register malicious agents.
      * **Improper Certificate Revocation (High):** Failure to immediately revoke the certificate of a compromised agent allows it to maintain trusted status.
    - **Cross-Advantages:** A strong PKI foundation secures all subsequent communication.
  + **Task: Logic Pipe Decision-Making:**
    - **Tools:** Rule engine (pyDatalog), ML Models (Reinforcement Learning).
    - **Risks:**
      * **Logic Loop Injection (High):** An attacker crafts an event that creates a self-perpetuating cycle (e.g., Red scan -> Blue alert -> Red scan of alert source -> Blue alert...). This is a resource exhaustion/DoS attack.
      * **ML Model Poisoning/Evasion (Critical):** If using an ML model for decisions, an attacker could feed it specifically crafted data (adversarial examples) to cause it to make catastrophic decisions. Poisoning the training data is also a major threat.
    - **Cross-advantages:** An ML-driven logic pipe can adapt to novel threats far faster than a static rule engine.

#### 2. Shared Knowledge Base (KB)

* **Overall Risk Level:** **Critical**
* **Rationale:** The KB is the swarm's source of truth. Poisoned data means the entire swarm operates on a false reality.
* **Micro-Detailed Task Analysis:**
  + **Task: Data Ingestion & Validation:**
    - **Tools:** Database APIs (pymongo, psycopg2), Data validation libraries (Pydantic).
    - **Risks:**
      * **Data Poisoning (Critical):** A single compromised agent writing false data (e.g., "Port 445 is closed" when it's open) can blindside the entire swarm.
      * **NoSQL Injection (High):** If using a NoSQL DB like MongoDB, improperly sanitized queries from an agent could lead to data exfiltration or corruption.
      * **Lack of Corroboration (High):** Believing the first report without seeking confirmation from another agent type is a major logical flaw.
    - **Cross-advantages:** A well-structured KB allows for powerful cross-correlation. A Threat Hunter can query for all activity related to an IOC found by the Network Monitor.

### II. Red Team Agent Risk Analysis

#### 1. Discovery Agent

* **Overall Risk Level:** **Medium**
* **Micro-Detailed Task Analysis:**
  + **Task network\_scan:**
    - **Tool:** nmap -sn or masscan --ping
    - **Risk:** Can generate significant network noise, alerting a sophisticated Blue Team. Masscan can be overly aggressive and crash fragile network devices.
    - **Advantage:** Fast way to map live hosts as targets for other agents.
  + **Task port\_scan / service\_enum:**
    - **Tool:** nmap -sV -O --script=default
    - **Risk (Medium):** Nmap Scripting Engine (NSE) scripts can be unstable, crashing the service they are probing or the agent's OS if there's a memory leak. Service banners can be spoofed to mislead the agent.
    - **Advantage:** Version information (-sV) is critical input for the Vulnerability Scanner. OS info (-O) helps the Exploitation Agent select a payload.

#### 2. OSINT Agent

* **Overall Risk Level:** **Low**
* **Micro-Detailed Task Analysis:**
  + **Task osint\_collection:**
    - **Tool:** theHarvester, Maltego queries, Shodan API
    - **Risk (Low):** Relies on third-party APIs. Rate limiting or API changes can disable the agent. Data can be intentionally poisoned (e.g., fake employee profiles on LinkedIn).
    - **Advantage:** Provides credentials, subdomains, and potential targets without touching the target network, offering a stealthy start.

#### 3. Vulnerability Scanner Agent

* **Overall Risk Level:** **High**
* **Micro-Detailed Task Analysis:**
  + **Task vuln\_scan:**
    - **Tool:** Nessus or OpenVAS
    - **Risk (High):** A motivated attacker could create a custom service that exploits a vulnerability in the *scanner itself* when it connects, leading to agent compromise. Scans can be resource-intensive and sometimes disruptive to production services. Output parsing (XML, JSON) is a major attack surface; a malformed report could exploit a parser bug.
    - **Advantage:** The primary source of actionable intelligence for the Exploitation agent. Automates a hugely time-consuming process.

#### 4. Exploitation Agent

* **Overall Risk Level:** **Critical**
* **Micro-Detailed Task Analysis:**
  + **Task execute\_exploit:**
    - **Tool:** Metasploit Framework (msfconsole)
    - **Risk (Critical):** This agent holds the "keys to the kingdom." If compromised, an attacker can use its capabilities (and credentials/sessions) to attack any target. An error in its logic could attack the wrong machine or use a destructive exploit in a production environment.
    - **Advantage:** The agent that actually achieves the objective of testing if a vulnerability is exploitable.

#### 5. Strategy Adaptation Agent

* **Overall Risk Level:** **High**
* **Micro-Detailed Task Analysis:**
  + **Task adapt\_attack\_strategy:**
    - **Tool:** BloodHound data, Reinforcement Learning models, Graph Database queries.
    - **Risk (High):** This agent's logic is complex. It could be tricked by a sophisticated defense into adapting to a less effective strategy or getting stuck in a logical loop. If its model is poisoned, it will consistently make poor strategic choices.
    - **Advantage:** This agent gives the swarm its "intelligence," allowing it to bypass static, rule-based defenses.

### III. Blue Team Agent Risk Analysis

#### 1. Network Monitor Agent

* **Overall Risk Level:** **High**
* **Micro-Detailed Task Analysis:**
  + **Task intrusion\_detection:**
    - **Tool:** Suricata, Zeek
    - **Risk (High):** Signature-based detection (Suricata) can be bypassed with novel or polymorphic attacks. An attacker can intentionally trigger low-level alerts to create "alert fatigue," hiding a real attack in the noise. Both tools have complex parsers for protocols; a malformed packet could crash the sensor.
    - **Advantage:** Provides real-time network visibility, the first line of defense. Zeek provides rich logs essential for the Threat Hunter.

#### 2. Log Analysis Agent

* **Overall Risk Level:** **High**
* **Micro-detailed Task Analysis:**
  + **Task log\_anomaly\_detection:**
    - **Tool:** Splunk, ELK Stack queries
    - **Risk (Medium):** Log sources can be compromised to stop sending logs or to send falsified logs. Centralized logging creates a single point of failure; if the logging pipeline breaks, this agent is blind.
    - **Advantage:** Connects disparate events from different systems to build a picture of an attack chain.

#### 3. Threat Hunter Agent

* **Overall Risk Level:** **Medium**
* **Micro-Detailed Task Analysis:**
  + **Task ioc\_scan:**
    - **Tool:** YARA, Sigma rule engines on EDR data.
    - **Risk (Medium):** Relies heavily on the quality of its rules and the integrity of the data it's searching (EDR logs). A poorly written YARA rule can cause massive performance issues on endpoints.
    - **Advantage:** Can find threats that other automated systems miss by using human-like hypothesis-driven investigation methods.

#### 4. Patch Management Agent

* **Overall Risk Level:** **Critical**
* **Micro-Detailed Task Analysis:**
  + **Task remediate\_vuln:**
    - **Tool:** Ansible, Puppet, WSUS
    - **Risk (Critical):** This agent has high-level administrative privileges on many systems. If compromised, an attacker can use it to push malicious updates or configuration changes, effectively owning the entire fleet of managed systems. A bad patch could also cause widespread outages.
    - **Advantage:** Automates a critical but often slow and error-prone defensive action.

#### 5. Containment Agent

* **Overall Risk Level:** **Critical**
* **Micro-Detailed Task Analysis:**
  + **Task network\_isolate:**
    - **Tool:** Firewall APIs, NAC APIs, EDR agent APIs.
    - **Risk (Critical):** Has the power to take systems offline. If tricked by a false positive, it could isolate a mission-critical server, causing a self-inflicted DoS. An attacker compromising this agent could isolate the entire security team from the network they are trying to defend.
    - **Advantage:** Provides the crucial "big red button" to stop an attack in its tracks and limit the blast radius.

## Appendix C: MCP Server Configuration & Agent Manifests

### Introduction

To move from a conceptual blueprint to an implementable system, we must define the concrete configurations that govern the swarm. This section outlines the server-side configuration for the Orchestrator (acting as a Master Control Program or MCP) and the standardized manifests that each agent uses to join and operate within the swarm.

### I. Orchestrator (MCP) Server Configuration

This is a conceptual config.yaml file for the main Orchestrator server. It defines its listening ports, security parameters, and connections to other core components.

# Orchestrator (MCP) Server Configuration: orchestrator-config.yaml

server:

# Interface for agents to connect for tasking and reporting

agent\_listener:

host: "0.0.0.0" # Listen on all interfaces

port: 4433

protocol: "https"

# Security configuration for the agent listener

security:

# Use mutual TLS to ensure only authenticated agents can connect

mTLS\_enabled: true

# Path to the private Certificate Authority that signs all valid agent certs

ca\_cert\_path: "/etc/orchestrator/pki/ca.crt"

# Path to the server's own certificate and private key

server\_cert\_path: "/etc/orchestrator/pki/server.crt"

server\_key\_path: "/etc/orchestrator/pki/server.key"

# List of revoked agent certificate serial numbers

crl\_path: "/etc/orchestrator/pki/revoked\_agents.crl"

# Connection details for the Shared Knowledge Base

knowledge\_base:

type: "elasticsearch" # or "mongodb", "postgresql"

host: "kb.internal.swarm"

port: 9200

credentials\_secret\_name: "kb-credentials" # Reference to a secret management system

# Logic Pipe configuration

logic\_pipe:

# Parameters to control the operational tempo and prevent DoS attacks

tempo\_control:

# Max tasks a single Red Team agent can execute per minute

red\_team\_max\_tasks\_per\_minute: 10

# Cooldown period (in seconds) before re-tasking an agent for the same target

target\_cooldown\_seconds: 300

# Confidence score threshold required to trigger a high-impact Blue Team action (e.g., containment)

high\_impact\_confidence\_threshold: 0.9

# API endpoints for agent communication

api\_endpoints:

register: "/api/v1/register"

heartbeat: "/api/v1/heartbeat"

request\_task: "/api/v1/task/request"

submit\_report: "/api/v1/report/submit"

### II. Standard Agent Manifest Template

Each agent, upon initialization, must load a manifest file. This manifest defines its identity, capabilities, and how it communicates with the MCP. This allows for flexible deployment and management of agents.

# Standard Agent Manifest Template: agent-manifest.yaml

# --- AGENT IDENTITY ---

agent\_id: "uuid-goes-here" # A unique identifier for this specific agent instance

agent\_name: "Human Readable Name" # e.g., "Nmap Scanner - DC-01"

agent\_type: "AgentClassType" # The class or type of the agent

team: "red | blue" # 'red' or 'blue'

# --- CAPABILITIES ---

# List of specific tasks this agent is authorized to perform

supported\_tasks:

- "task\_name\_1"

- "task\_name\_2"

# --- C2 CONFIGURATION ---

c2\_server:

host: "orchestrator.internal.swarm"

port: 4433

protocol: "https"

# --- AGENT SECURITY ---

security:

client\_cert\_path: "./pki/agent.crt"

client\_key\_path: "./pki/agent.key"

# --- OPERATIONAL PARAMETERS ---

heartbeat\_interval\_seconds: 60

task\_timeout\_seconds: 1800

### III. Example Agent Manifests (Expanded)

This section provides a complete set of example manifests for every agent defined in the blueprint.

#### Red Team Manifests

**1. Discovery Agent**

agent\_id: "a1b2c3d4-discovery-01"

agent\_name: "Nmap & Impacket Scanner - DC-01"

agent\_type: "DiscoveryAgent"

team: "red"

supported\_tasks:

- "network\_scan"

- "port\_scan"

- "credential\_gathering" # For Kerberoasting, etc.

c2\_server:

host: "orchestrator.internal.swarm"

port: 4433

security:

client\_cert\_path: "./pki/discovery01.crt"

client\_key\_path: "./pki/discovery01.key"

heartbeat\_interval\_seconds: 60

task\_timeout\_seconds: 3600 # Nmap scans can be long

**2. OSINT Agent**

agent\_id: "b2c3d4e5-osint-01"

agent\_name: "Public Reconnaissance Unit"

agent\_type: "OSINTAgent"

team: "red"

supported\_tasks:

- "osint\_collection"

- "domain\_analysis"

- "cert\_log\_analysis"

c2\_server:

host: "orchestrator.internal.swarm"

port: 4433

security:

client\_cert\_path: "./pki/osint01.crt"

client\_key\_path: "./pki/osint01.key"

heartbeat\_interval\_seconds: 120

task\_timeout\_seconds: 900

**3. Vulnerability Scanner Agent**

agent\_id: "c3d4e5f6-vulnscan-01"

agent\_name: "Nessus Scanner - Production Web"

agent\_type: "VulnerabilityScanner"

team: "red"

supported\_tasks:

- "vuln\_scan"

- "config\_audit"

- "webapp\_scan"

c2\_server:

host: "orchestrator.internal.swarm"

port: 4433

security:

client\_cert\_path: "./pki/vulnscan01.crt"

client\_key\_path: "./pki/vulnscan01.key"

heartbeat\_interval\_seconds: 60

task\_timeout\_seconds: 7200 # Vulnerability scans are very slow

**4. Exploitation Agent**

agent\_id: "d4e5f6g7-exploit-01"

agent\_name: "Metasploit & C3 Operator"

agent\_type: "ExploitationAgent"

team: "red"

supported\_tasks:

- "execute\_exploit"

- "deliver\_payload"

- "c3\_c2\_channel"

c2\_server:

host: "orchestrator.internal.swarm"

port: 4433

security:

client\_cert\_path: "./pki/exploit01.crt"

client\_key\_path: "./pki/exploit01.key"

heartbeat\_interval\_seconds: 30 # Needs responsive C2

task\_timeout\_seconds: 1800

**5. Strategy Adaptation Agent**

agent\_id: "e5f6g7h8-strategy-01"

agent\_name: "AI Strategy Engine"

agent\_type: "StrategyAdaptationAgent"

team: "red"

supported\_tasks:

- "adapt\_attack\_strategy"

- "generate\_tree\_of\_attacks"

- "re\_evaluate\_target"

c2\_server:

host: "orchestrator.internal.swarm"

port: 4433

security:

client\_cert\_path: "./pki/strategy01.crt"

client\_key\_path: "./pki/strategy01.key"

heartbeat\_interval\_seconds: 300

task\_timeout\_seconds: 600

#### Blue Team Manifests

**1. Network Monitor Agent**

agent\_id: "f6g7h8i9-netmon-01"

agent\_name: "Suricata Sensor - EdgeFirewall"

agent\_type: "NetworkMonitorAgent"

team: "blue"

supported\_tasks:

- "traffic\_analysis"

- "intrusion\_detection"

c2\_server:

host: "orchestrator.internal.swarm"

port: 4433

security:

client\_cert\_path: "./pki/netmon01.crt"

client\_key\_path: "./pki/netmon01.key"

heartbeat\_interval\_seconds: 30

task\_timeout\_seconds: 86400 # Long-running monitoring task

**2. Log Analysis Agent**

agent\_id: "g7h8i9j0-logalyze-01"

agent\_name: "ELK Stack Correlator"

agent\_type: "LogAnalysisAgent"

team: "blue"

supported\_tasks:

- "log\_collection"

- "log\_parsing"

- "log\_anomaly\_detection"

c2\_server:

host: "orchestrator.internal.swarm"

port: 4433

security:

client\_cert\_path: "./pki/logalyze01.crt"

client\_key\_path: "./pki/logalyze01.key"

heartbeat\_interval\_seconds: 60

task\_timeout\_seconds: 86400

**3. Threat Hunter Agent**

agent\_id: "h8i9j0k1-hunter-01"

agent\_name: "EDR & YARA Hunter"

agent\_type: "ThreatHunterAgent"

team: "blue"

supported\_tasks:

- "proactive\_monitor"

- "behavioral\_analysis"

- "ioc\_scan"

c2\_server:

host: "orchestrator.internal.swarm"

port: 4433

security:

client\_cert\_path: "./pki/hunter01.crt"

client\_key\_path: "./pki/hunter01.key"

heartbeat\_interval\_seconds: 120

task\_timeout\_seconds: 3600

**4. Patch Management Agent**

agent\_id: "i9j0k1l2-patch-01"

agent\_name: "Ansible Patch Runner"

agent\_type: "PatchManagementAgent"

team: "blue"

supported\_tasks:

- "remediate\_vuln"

- "enforce\_config"

c2\_server:

host: "orchestrator.internal.swarm"

port: 4433

security:

client\_cert\_path: "./pki/patch01.crt"

client\_key\_path: "./pki/patch01.key"

heartbeat\_interval\_seconds: 300

task\_timeout\_seconds: 7200 # Patching can take time

**5. Containment Agent**

agent\_id: "j0k1l2m3-contain-01"

agent\_name: "Firewall & EDR Containment"

agent\_type: "ContainmentAgent"

team: "blue"

supported\_tasks:

- "network\_isolate"

- "process\_terminate"

- "block\_ip\_domain"

c2\_server:

host: "orchestrator.internal.swarm"

port: 4433

security:

client\_cert\_path: "./pki/contain01.crt"

client\_key\_path: "./pki/contain01.key"

heartbeat\_interval\_seconds: 10 # Needs to be highly responsive

task\_timeout\_seconds: 300

**6. AI Monitoring Agent**

agent\_id: "k1l2m3n4-aimon-01"

agent\_name: "Orchestrator Watcher"

agent\_type: "AIMonitoringAgent"

team: "blue" # Aligned with defense

supported\_tasks:

- "monitor\_reasoning\_chain"

- "prompt\_sanity\_check"

c2\_server:

host: "orchestrator.internal.swarm"

port: 4433

security:

client\_cert\_path: "./pki/aimon01.crt"

client\_key\_path: "./pki/aimon01.key"

heartbeat\_interval\_seconds: 5 # Must be very fast

task\_timeout\_seconds: 10