🕸️ Tools for Implementing the Autonomous Deception Mesh (The Mash)

This document outlines the exact tools and supporting technologies required to build the Autonomous Deception Mesh (Mash) into the Unified OT Ransomware Protection Platform. These tools are compatible with Python, containerized microservices, and industrial OT protocols like Modbus and OPC-UA.

# 🎯 1. Creating and Managing Honeypots

• Tool: Conpot (open-source ICS honeypot) / Custom Virtual Honeypot Frameworks (Python, Go)  
  
Use these tools to simulate OT devices (e.g., PLCs, RTUs) on the network. Emulate industrial protocols (Modbus, DNP3, etc.) and deploy across VLANs. These honeypots collect attacker interactions while appearing indistinguishable from real devices.

# 🤖 2. Dynamic Deception Orchestration

• Tool: Custom Python-based deception manager (or built-in in Edge Node)  
  
- Controls which honeypots to deploy dynamically.  
- Uses AI inputs to vary deception behavior over time (e.g., IP changes, file names).  
- Integrates with SOAR for automated responses when deception assets are triggered.

# 📦 3. Packet Logging and Interaction Capture

• Tool: tcpdump, Wireshark, or Zeek (formerly Bro)  
  
Used for full-packet capture of honeypot interactions. All traffic toward deception nodes should be logged and flagged for forensic analysis. This also helps enrich threat intelligence.

# 📄 4. Log Integrity and Analysis

• Tools: ELK Stack (Elasticsearch, Logstash, Kibana), Wazuh (optional)  
  
- Honeypot logs are forwarded to ELK.  
- Kibana dashboards display interaction heatmaps, attack vectors.  
- Wazuh (if used) can alert on specific attacker behaviors in honeypots.

# 🧠 5. Threat Intelligence Correlation

• Tool: MISP (Malware Information Sharing Platform) / Threat Intelligence Hub (Custom)  
  
- Deception logs are analyzed and correlated with known TTPs (MITRE ATT&CK).  
- Unique attacker behaviors are shared back to the federated model to immunize other sites.

# 📍 6. Edge Deployment and Isolation

• Tool: Docker or Podman for containerization + K3s (lightweight Kubernetes)  
  
- Honeypots and deception orchestrators run as microservices.  
- Can isolate malicious IPs or interfaces instantly.

# 🔐 7. Securing Deception Infrastructure

• Tools: TLS 1.3, Python cryptography, Mutual gRPC, X.509 certs  
  
- All deception telemetry is encrypted during transit.  
- Internal microservice comms (e.g., deception manager ↔ orchestrator) use gRPC + mTLS.  
- Secrets (e.g., decoy credentials) are stored via HashiCorp Vault.

# 🖥️ Recommended Tool Stack

|  |  |  |
| --- | --- | --- |
| Functionality | Tool from Logging Doc | How It Applies to the Mash |
| Encrypt Data at Rest | cryptography (AES-256) | Encrypt logs, attacker artifacts, and decoy credential files. |
| Secure Data in Transit | TLS 1.3 | Protect telemetry and inter-service communication. |
| Verify Message Integrity | Python hmac module | Ensure attacker telemetry from honeypots hasn't been altered. |
| Logging System Activity | logging (Python built-in) | Log all interactions with decoys and system events. |
| Visualizing and Managing Logs | ELK Stack / Wazuh | Track honeypot hits, source IPs, and alert trends. |
| Managing Secrets and Keys | HashiCorp Vault / .env files | Store honeypot secrets and configuration securely. |
| Securing MQTT Communication | Mosquitto with TLS | Secure MQTT if used for telemetry or control messages. |