# Automated Incident Response & Containment - Full Implementation Plan

## 1. Overview

This module is designed to react to ransomware threats and cyberattacks automatically, minimizing response time and limiting damage in Operational Technology (OT) environments. It integrates tightly with a custom-built AI detection engine and other modules such as Secure Backup, Logging, and Admin Notification.

## 2. Key Capabilities and Implementation Steps

### 2.1 Ransomware Detection Triggers Response

Use a trained machine learning model (e.g., RandomForestClassifier or LSTM) to analyze file behavior, CPU usage, and network traffic in real time. Upon detecting ransomware behavior, the AI triggers incident response functions.  
- Language: Python  
- Libraries: sklearn, pandas, watchdog (for real-time monitoring)

### 2.2 Immediate Isolation of Infected Systems

Use Python to execute OS-level commands to isolate the infected node:  
- Disconnect network interface using `os.system("ifconfig eth0 down")`  
- Kill malicious processes using `psutil` or `subprocess`.  
- Mark system status as 'isolated' in the system state file.

### 2.3 Automated Playbooks

Design Python scripts that act as playbooks to:  
- Pause critical processes (e.g., SCADA or PLC control)  
- Create just-in-time backups using `shutil.copytree()`  
- Lock down user access by disabling accounts or using firewall rules.

### 2.4 Admin Notification System

Use SMTP in Python to send email alerts with details:  
- Device ID, detected behavior, timestamp, file hash, and threat vector.  
- Use the `smtplib` and `email` libraries.

### 2.5 Smart Backup & Restore Triggers

Implement file snapshot system:  
- Regularly save clean versions of critical config and data files.  
- Restore from these when ransomware is detected.  
- Use `os`, `shutil`, or `rsync` for fast file operations.

### 2.6 Secure Logging

Implement tamper-proof logging:  
- Write logs to a file in append-only mode.  
- Use HMAC (from hashlib) to hash log entries for verification.  
- Log all actions: detections, responses, notifications, backups.

### 2.7 Admin Dashboard (Optional)

Use Flask or Django to create a lightweight web dashboard:  
- Show system status, recent alerts, and action logs.  
- Allow authorized control (e.g., force backup or isolation).

## 3. Integration Flow (Described)

[Ransomware Detection Engine]  
 ↓  
[SOAR Engine] → [Automated Response Playbook]  
 ↓ ↓  
 [Secure Logging] [Backup & Restore Trigger]  
 ↓ ↓  
 [Alert Admins] [System Rollback]

## 4. Technologies and Tools Used

- Language: Python  
- ML Libraries: sklearn, pandas, numpy  
- Monitoring: watchdog, psutil  
- Response & OS Control: os, subprocess, shutil  
- Logging: logging, hashlib (HMAC)  
- Email: smtplib, email  
- Web Dashboard: Flask (optional)

## 5. Summary

This implementation ensures a complete, self-contained security loop capable of responding to ransomware threats in real-time. The system acts automatically, leveraging AI detection, secure backups, and automated OS-level responses to neutralize threats without relying on third-party SOAR software.