Security Monitoring and Incident Response: Detailed Report

Project Goal: To demonstrate security monitoring knowledge through practical implementation, including setting up basic security monitoring, detailing a use case, and documenting an incident response scenario.

I. Basic Security Monitoring Setup

For this mock scenario, we'll set up basic security monitoring using a combination of tools and techniques.

A. Components

1. **Log Management:**
    - We'll use an ELK stack (Elasticsearch, Logstash, Kibana) to collect, process, and visualize logs from various sources (web servers, firewalls, operating systems).
        - Elasticsearch*: A NoSQL database to store logs.
        - Logstash*: A data processing pipeline to ingest, parse, and transform logs.
        - Kibana*: A visualization tool to create dashboards and alerts.
2. **Intrusion Detection System (IDS):**
    - We'll deploy a host-based intrusion detection system (HIDS) like OSSEC on critical servers to monitor file integrity, detect rootkits, and alert on suspicious activity.
3. **Network Intrusion Detection System (NIDS):**
    - We'll use Snort, a network intrusion detection and prevention system, to monitor network traffic for malicious patterns and known attack signatures.

B. Configuration

1. **ELK Stack Configuration:**
        - Logstash Configuration*: Configure Logstash to receive logs from web servers (e.g., Apache access logs), firewalls (e.g., iptables logs), and OSSEC alerts. Define filters to parse the logs and extract relevant fields (e.g., IP addresses, timestamps, user agents, event types).
        - Elasticsearch Configuration*: Configure Elasticsearch to create indices for the logs and optimize storage.

- Kibana Configuration*: Create a Kibana dashboard to visualize the logs, showing key metrics like traffic volume, error rates, and security events. Set up alerts to trigger when specific events occur (e.g., excessive login failures, suspicious network connections).

2. **OSSEC Configuration:**
    - Install the OSSEC agent on the servers to be monitored.
    - Configure OSSEC to monitor critical files and directories for changes (e.g., `/etc/passwd`, `/var/www/html`), detect rootkits, and generate alerts for suspicious system calls.
    - Send OSSEC alerts to Logstash for centralized logging and analysis.
3. **Snort Configuration:**
    - Install Snort on a network gateway or sensor.
    - Configure Snort rules to detect known attack patterns (e.g., SQL injection attempts, port scans, denial-of-service attacks).
    - Integrate Snort with a unified security management console.

C. Use Case: Detecting a Web Server Attack

1. **Detection Rules:**
    - Snort Rule*: Create a Snort rule to detect SQL injection attempts in HTTP requests:
      ```
      alert tcp any any -> any 80 (msg:"WEB-SQLI
      SQL Injection Attempt";
      flow:established,to_server; content:"SELECT";
      http_uri; content:"' OR '1'='1"; http_uri;
      nocase; classtype:web-application-attack;
      sid:300001; rev:1;)
      ```
    -
    -
    - Kibana Alert*: Create a Kibana alert to detect an unusual number of 500 errors on the web server within a short period (e.g., more than 100 errors in 5 minutes), which could indicate a web application attack. The query would look for entries with `response_code:500` in the web server logs.
2. **Alert Prioritization:**

- Snort Alert Priority*: The Snort rule above is classified as a "web-application-attack," which would be assigned a high priority.
- Kibana Alert Priority*: The Kibana alert for excessive 500 errors would also be considered high priority, as it suggests a potential attack or server compromise.
- Prioritization Process*:
  - Alerts are prioritized based on severity (e.g., critical, high, medium, low) using a combination of factors:
    - CVSS score of the underlying vulnerability.
    - Type of attack.
    - Target system.
    - Potential impact.
  - A security analyst reviews the alerts and adjusts the priority if necessary.

3. **Response Procedures:**

- When a high-priority alert is triggered (e.g., a SQL injection attempt is detected by Snort or excessive 500 errors are detected by Kibana):
  - **Initial Analysis**: The security analyst investigates the alert to confirm its validity and determine the scope of the attack.
  - **Containment**:
    - If a SQL injection is confirmed, the web server is isolated to prevent further database compromise. Firewall rules are updated to block traffic from the attacker's IP address.
    - If excessive 500 errors indicate a potential denial-of-service attack, traffic shaping or blacklisting is implemented.
  - **Eradication**:
    - For a SQL injection, the vulnerable code is patched, and the web server is hardened. The database is checked for unauthorized modifications, and backups are restored if necessary.

        - For a denial-of-service attack, the attacker's source is blocked, and mitigation measures (e.g., DDoS protection services) are implemented.

    - **Recovery**: The web server is brought back online after patching and hardening. Systems are monitored closely for any further suspicious activity.

    - **Post-Incident Activity**: The incident is documented, and lessons learned are incorporated into security monitoring and incident response procedures.

II. Incident Response Scenario

A. Scenario: Ransomware Attack

1. **Incident Description**:
    - Multiple workstations in the finance department are encrypted with ransomware. Users report that their files have been renamed with a `.locked` extension, and a ransom note demands Bitcoin payment for decryption.
2. **Classification of Incident**:
        - Type*: Ransomware attack.
        - Category*: Malware incident.
        - Impact*: High (loss of data availability, potential data breach).
        - Priority*: Critical.

B. Response Steps Taken

1. **Preparation**:
    - The incident response team was activated according to the incident response plan.
    - Communication channels were established (secure chat room, conference call).
2. **Identification**:
    - The ransomware variant was identified through analysis of the ransom note and encrypted files.
    - Affected systems were identified and isolated from the network to prevent further spread of the infection.

- Log analysis (from SIEM) was used to determine the initial infection vector (e.g., phishing email, malicious download).

3. **Containment**:
    - Affected workstations were disconnected from the network.
    - Network shares were disabled to prevent the ransomware from spreading to other systems.
    - Firewall rules were updated to block communication with known C2 servers associated with the ransomware.
4. **Eradication**:
    - Infected workstations were reimaged from clean backups.
    - Antivirus software was updated with the latest signatures to detect and remove the ransomware.
    - A vulnerability scan was performed to identify and patch any systems with similar vulnerabilities.
5. **Recovery**:
    - Data was restored from backups.
    - Systems were brought back online after verifying that they were clean.
    - Users were provided with training on how to identify and avoid phishing emails and other social engineering attacks.
6. **Post-Incident Activity**:
    - A post-incident report was created, documenting the incident details, response steps, and lessons learned.

C. Lessons Learned

- **Importance of Backups**: The incident highlighted the critical importance of having up-to-date and tested backups. The ability to restore data from backups minimized the impact of the ransomware attack.
- **User Awareness Training**: The incident emphasized the need for regular user awareness training to educate employees about phishing and other social engineering tactics.
- **Vulnerability Management**: The incident underscored the importance of keeping systems patched and up-to-date to prevent malware infections.

- **Incident Response Plan**: The pre-existing incident response plan facilitated a coordinated and effective response to the attack. Regular testing and updating of the plan are essential.
- **SIEM Value**: The Security Information and Event Management (SIEM) system played a crucial role in log analysis, helping to identify the source of the attack and track its spread.