***4.1 Overview of Nessus***

### Understanding Nessus

#### What is Nessus?

Nessus is a **vulnerability assessment tool** developed by Tenable. It is designed to identify security vulnerabilities, misconfigurations, and compliance issues across various IT environments. It scans networks, applications, and systems for known vulnerabilities, generating detailed reports that help security teams take preventive action.

#### Key Features of Nessus

1. **Comprehensive Vulnerability Scanning** – Nessus scans IT assets for known vulnerabilities, misconfigurations, and compliance issues across multiple platforms, including Windows, Linux, and cloud environments.
2. **Real-Time Threat Intelligence** – It integrates with live threat intelligence feeds to detect emerging vulnerabilities.
3. **Automated Security Audits** – Nessus performs security assessments based on industry standards (e.g., CIS, NIST, ISO 27001).
4. **Detailed Reporting & Risk Analysis** – Generates reports that prioritize vulnerabilities based on severity, exploitability, and impact.
5. **Integration with Incident Response Systems** – Nessus can be integrated with SIEM (Security Information and Event Management) tools, ticketing systems, and response platforms for automated mitigation.

### Relevance of Nessus to Swift Incidence Response

**Swift Incidence Response** is designed to detect, manage, and respond to security incidents efficiently. Nessus aligns with this goal by providing a **proactive** approach to security through vulnerability assessments. Here’s how Nessus can enhance Swift Incidence Response:

1. **Early Detection of Vulnerabilities**
   * By running Nessus scans periodically, Swift Incidence Response can **identify security weaknesses before they are exploited**. This helps in reducing the risk of incidents caused by known vulnerabilities.
2. **Automated Incident Creation**
   * Detected vulnerabilities can be **automatically logged as security incidents** in the Swift Incidence Response system. This ensures that vulnerabilities are tracked and resolved systematically.
3. **Risk-Based Prioritization**
   * Nessus assigns a **risk score** to each vulnerability based on CVSS (Common Vulnerability Scoring System). Swift Incidence Response can leverage this data to **prioritize high-risk incidents** and allocate resources accordingly.
4. **Remediation and Patch Management**
   * The tool provides **detailed remediation steps** for each vulnerability. Swift Incidence Response can automate notifications and assign tasks to security teams for **patching affected systems**.
5. **Integration with Threat Intelligence**
   * By integrating Nessus with **external threat intelligence platforms**, Swift Incidence Response can **correlate vulnerabilities with active exploits in the wild**, allowing security teams to take preemptive action.
6. **Compliance Monitoring & Auditing**
   * Organizations following compliance frameworks (e.g., PCI DSS, HIPAA, ISO 27001) can use Nessus for **continuous monitoring and auditing**. Swift Incidence Response can maintain logs of vulnerability assessments to ensure compliance with regulatory requirements.

***4.2 Proposed Solution***

**1. SSL Certificate Cannot Be Trusted**

✅ **Solution**:

* Obtain a **valid SSL certificate** from a trusted Certificate Authority (CA) (e.g., Let's Encrypt, DigiCert).
* Use **Certbot** for automated renewal.
* Enforce **TLS 1.2+** and disable older protocols.
* Implement **HSTS (HTTP Strict Transport Security)** in NGINX/Apache.

**2. Node.js Version-Related Vulnerabilities**

✅ **Solution**:

* **Upgrade Node.js** to the latest stable LTS version (>= 21.7.2).
* Use **NVM (Node Version Manager)** for version control.
* Run **npm audit fix** to patch dependencies.
* Secure Node.js apps with:
  + **Helmet.js** (secures HTTP headers).
  + **CORS policy enforcement**.
  + **Rate limiting (express-rate-limit)** to prevent DoS attacks.
  + **CSRF protection (csurf middleware)**.

**3. OpenJDK Security Vulnerabilities**

✅ **Solution**:

* **Upgrade OpenJDK** to the latest version (≥ 23.0.2).
* Use **SDKMAN!** for JDK version management.
* Enable security flags (-Dcom.sun.net.ssl.enableECC=true).
* Run **OWASP Dependency-Check** to scan for library vulnerabilities.

**4. Apache HTTP Server Site Enumeration**

✅ **Solution**:

* **Disable directory listing** in Apache config:

apache

CopyEdit

Options -Indexes

* Use .htaccess rules to restrict access.
* Enable **ModSecurity (WAF)** for request filtering.
* Implement **Fail2Ban** to block repeated malicious requests.

**5. Continuous Security Monitoring & DevSecOps**

✅ **Solution**:

* Implement **SIEM tools** for real-time monitoring (Splunk / Wazuh / ELK Stack).
* Use **OWASP ZAP / Nmap** for automated vulnerability scanning.
* Enable **GitHub Dependabot / Trivy** for dependency security in CI/CD pipelines.
* Conduct **regular penetration testing** to identify new vulnerabilities.