Vulnerability assessment task

What was the device I used as a target

- **It was a metasploitable machine that I installed on my device and the task have been done from my kali linux machine targeting the metasploitable machine .**

How did I started

- **Scanned the network to identify the ip of the metasploitable machine to lock in**
- **I used 4 types of scans all are explained on the github repo**
- **The result of the scans are also uploaded to the repo**

▼ What are the findings

▼ ftp-vsftpd-backdoor

- on port **21** found a **backdoor** that can be exploited
- **Reported** on 04-07-2011 and **disclosed** on 03-07-2011
- **Severity** : **Medium**

▼ SSL poodle

- An **SSL** poodle information leak
- The **SSL** protocol 3.0, as used in OpenSSL through 1.0.1i and other products, uses nondeterministic CBC padding, which makes it easier for man-in-the-middle attackers to obtain cleartext data via a padding-oracle attack, aka the "POODLE" issue.
- **Disclosed** on 14-10-2014
- **Severity Level**: **High**
- **Recommendation**: Disable SSL 3.0 on all servers and clients. Ensure that the system supports only secure versions of TLS (preferably TLS

1.2 or higher). This will mitigate the risk associated with the POODLE vulnerability.

▼ ssl-dh-param

- Anonymous Diffie-Hellman Key Exchange that cause a **MitM** Vulnerability
- Transport Layer Security (TLS) services that use anonymous Diffie-Hellman key exchange only provide protection against passive eavesdropping, and are vulnerable to active man-in-the-middle attacks which could completely compromise the confidentiality and integrity of any data exchanged over the resulting session.
- **Severity Level**: **High**
- **Recommendations** : Immediate action should be taken to disable anonymous Diffie-Hellman key exchange and configure your TLS services to use strong, authenticated cipher suites. This will protect against active MITM attacks and ensure the confidentiality and integrity of your communications.

▼ Transport Layer Security (TLS) Protocol DHE_EXPORT Ciphers Downgrade MitM (Logjam)

- The Transport Layer Security (TLS) protocol contains a flaw that is triggered when handling Diffie-Hellman key exchanges defined with the DHE_EXPORT cipher. This may allow a man-in-the-middle attacker to downgrade the security of a TLS session to 512-bit export-grade cryptography, which is significantly weaker, allowing the attacker to more easily break the encryption and monitor or tamper with the encrypted stream.
- **Disclosed** on 19-05-2015
- **Severity Level**: **High**
- **Recommendations**: Immediate action is required to disable `DHE_EXPORT` ciphers and ensure that only strong, secure cipher suites are supported. This will protect against the Logjam attack and safeguard the confidentiality and integrity of your TLS sessions.

▼ Diffie-Hellman Key Exchange Insufficient Group Strength

- Transport Layer Security (TLS) services that use Diffie-Hellman groups of insufficient strength, especially those using one of a few commonly shared groups, may be susceptible to passive eavesdropping attacks.
- **Severity Level**: **Medium to High**
- **Recommendations**: Immediate steps should be taken to disable weak DH groups, enforce strong group sizes, and potentially move towards using elliptic curve cryptography. These actions will greatly reduce the risk of passive eavesdropping attacks.

▼ http-slowloris-check

- script used with Nmap to check for vulnerabilities related to the **Slowloris** attack. **Slowloris** is a type of **Denial-of-Service (DoS)** attack where the attacker sends partial **HTTP** requests to a web server at a very slow rate, keeping the connections open and eventually exhausting the server's resources, leading to a denial of service.
- **Disclosed on** 17-09-2009
- **Severity: Medium to high**
- **Recommendations:** Servers that are vulnerable to Slowloris attacks should be promptly secured to prevent potential disruptions. Implementing proper timeout settings, connection limits, and possibly deploying a WAF are crucial steps to mitigating this vulnerability

▼ possible CSRF vulnerabilities

- Path: http://192.168.18.44:80/dvwa/

  > Form id:
  > Form action: login.php

- Path: http://192.168.18.44:80/mutillidae/?page=user-info.php

> Form id: id-bad-cred-tr
> Form action: ./index.php?page=user-info.php

- Path: http://192.168.18.44:80/mutillidae/?page=login.php

  > Form id: idloginform
  > Form action: index.php?page=login.php

- Path: http://192.168.18.44:80/mutillidae/index.php?page=text-file-viewer.php

  > Form id: id-bad-cred-tr
  > Form action: index.php?page=text-file-viewer.php

- Path: http://192.168.18.44:80/mutillidae/?page=text-file-viewer.php

  > Form id: id-bad-cred-tr
  > Form action: index.php?page=text-file-viewer.php

▼ rmi-vuln-classloader

- Default configuration of RMI registry allows loading classes from remote URLs which can lead to remote code execution.
- **Severity: High**
- **Recommendations:**
  - **Restrict Network Access**: Limit access to the RMI service to trusted hosts using firewalls or security groups.
  - **Disable or Secure Class Loading**: Configure the RMI service to prevent loading arbitrary classes from remote sources.

- **Keep JVM Updated**: Ensure your Java Virtual Machine (JVM) is updated with the latest security patches.
- **Authenticate RMI Connections**: Implement authentication to prevent unauthorized access to the RMI service.
- **Use an RMI Security Manager**: Deploy an RMI Security Manager to control what classes can be loaded and executed.
- **Regularly Review RMI Configurations**: Periodically audit your RMI service settings to ensure they remain secure.

▼ ssl-ccs-injection

- OpenSSL before 0.9.8za, 1.0.0 before 1.0.0m, and 1.0.1 before 1.0.1h does not properly restrict processing of ChangeCipherSpec messages, which allows man-in-the-middle attackers to trigger use of a zero length master key in certain OpenSSL-to-OpenSSL communications, and consequently hijack sessions or obtain sensitive information, via a crafted TLS handshake, aka the "CCS Injection" vulnerability.
- **Severity: High**
- **Recommendations:**
- **Update OpenSSL**:
    - **Immediate Action**: Update OpenSSL to a version that is not affected by this vulnerability. Versions 1.0.1g and later include fixes for CVE-2014-0224.
    - Example for Ubuntu:

    ```
    bashCopy code
    sudo apt-get update
    sudo apt-get install --only-upgrade openssl
    ```

- **Patch and Update Systems**:
    - Ensure all systems and software that use OpenSSL are patched and updated to the latest versions to protect against this and other

vulnerabilities.

- **Monitor and Audit SSL/TLS Configurations**:
    - Regularly audit your SSL/TLS configurations using tools like SSL Labs’ SSL Test to ensure they adhere to best practices and are free from known vulnerabilities.
- **Implement Stronger Security Protocols**:
    - Consider disabling older, less secure protocols like SSLv2 and SSLv3, and ensure you are using strong cipher suites and configurations that mitigate the risk of various attacks.
- **Use Intrusion Detection Systems (IDS)**:
    - Deploy an IDS to monitor for signs of attempted exploitation of SSL/TLS vulnerabilities, including suspicious ChangeCipherSpec messages.