# Task 3:  Perform a Basic Vulnerability Scan on Your PC

## Vulnerabilities found in the scan report

### 1)**High Risk: SSL Medium Strength Cipher Suites Supported (SWEET32)**

What It Is?:
The SWEET32 vulnerability targets 64-bit block ciphers—most notably Triple DES (3DES) used in SSL/TLS. Under prolonged use (e.g., in long sessions), these ciphers can leak small amounts of plaintext, making it easier for an attacker on the same network to potentially decrypt sensitive data.

**Steps to Resolve (Windows Example):**

1. **Disable 3DES via the Registry:**
    - **Open the Registry Editor (regedit).**
    - **Navigate to:**

**```HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers\Triple DES 168 ```**

    - **If not already present, create a DWORD value named Enabled and set it to 0.**
2. **Disable Other Weak Ciphers (if applicable):**
    - **Similarly, navigate to related keys such as:**

**```HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers\DES 56/56 ```**

    - **Set the Enabled value to 0.**
3. **Restart the System/Services:**
    - **Reboot your machine or restart the relevant services to apply the changes.**
4. **Re-verify with a Scan:**
    - **Run Nessus again to ensure the vulnerable ciphers are no longer enabled.**

**2) Medium Risk: IP Forwarding Enabled**

What It Is:
Enabling IP forwarding on a device that isn't intended to act as a router can allow the system to forward network traffic. This misconfiguration may open avenues for attackers to intercept or redirect traffic, potentially bypassing firewall or network access controls.

Steps to Resolve:

- On Windows:
    - Open the Registry Editor (regedit) and navigate to:

```HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters```

    - Set the IPEnableRouter value to 0.
    - Restart your machine to apply the changes.

**3)Medium Risk: SSL Certificate cannot be Trusted**

What It Is:
This issue arises when your server presents an SSL certificate that isn't signed by a recognized Certificate Authority (CA) or has an incomplete certificate chain. Common causes include using a self-signed certificate or failing to include necessary intermediate certificates, which leads browsers to flag the certificate as untrusted.

**Steps to Resolve:**

1. **Obtain a Trusted Certificate:**
   Replace the current certificate with one issued by a reputable CA.
2. **Install the Full Certificate Chain:**
   Ensure that all intermediate certificates are correctly installed so that the chain of trust is complete and verifiable by client devices.
3. **Verify Certificate Details:**
    - Confirm that the common name (CN) or Subject Alternative Name (SAN) matches your server's hostname.
    - Check that the certificate has not expired or been revoked.
4. **Double-Check System Settings:**
   Make sure your system's date and time are accurate, as discrepancies can also cause trust errors.

By following these steps, you'll ensure that your SSL certificate is properly trusted by clients and browsers, eliminating warnings and securing your site's communications.
```

**4)Medium Risk: SSL Self-Signed Certificate**

What It Is:
A self-signed certificate isn't issued by a trusted Certificate Authority (CA), meaning users or systems won't inherently trust the SSL connection. This is acceptable only in testing environments, but in production it can leave you open to man-in-the-middle attacks.

**Steps to Resolve:**

1. **Replace with a Trusted Certificate:**
    - **Obtain a Certificate:** Acquire an SSL certificate from a reputable CA (such as Let's Encrypt, DigiCert, etc.).
    - **Install the Certificate:** Configure your web server with the new certificate along with the complete chain (server, intermediate, and root certificates).
2. **Alternatively, Trust the Self-Signed Certificate (Development Only):**
    - **Import into Trusted Store:** If you must use a self-signed certificate (e.g., in a testing environment), import it into your system's trusted certificate store.
        - On **Windows**, use the Certificate Manager (MMC) to add the certificate to "Trusted Root Certification Authorities."
        - On **Linux**, add the certificate to /usr/local/share/ca-certificates/ (update with sudo update-ca-certificates) or your distro's equivalent.
    - **Security Note:** This method should only be used in controlled environments, as it weakens the chain-of-trust in production.
3. **Verify the Configuration:**

- Test your server using an SSL checking tool or a web browser to confirm that the certificate is now trusted.

By following these steps, you ensure that your SSL communications are validated by trusted authorities, thereby enhancing the security of your SSL/TLS deployments.

**5)Low Risk: ICMP Timestamp Request Remote Date Disclosure**

What It Is:
This vulnerability (often referenced as CVE-1999-0524) occurs when a system responds to unsolicited ICMP timestamp requests (ICMP type 13) with timestamp replies (ICMP type 14). The disclosed time can help attackers refine their attacks—such as defeating time-based security mechanisms or gathering system intelligence.

**Steps to Rseolve:**

- **Block ICMP Timestamp Traffic via Firewall/ACLs:**
    - **On Linux:**
      Use iptables to drop ICMP timestamp messages:

      iptables -A INPUT -p icmp --icmp-type timestamp-request -j DROP iptables -A OUTPUT -p icmp --icmp-type timestamp-reply -j DROP

    - **On Windows:**
      Create inbound and outbound firewall rules to block ICMP types 13 and 14.
    - **On Cisco or Other Network Devices:**
      Configure an Access Control List (ACL) to deny ICMP timestamp requests and replies. For example:

  access-list 101 deny icmp any any echo-timestamp access-list 101 deny icmp any any timestamp-reply

- Then apply the ACL to the appropriate interface.

**Verify the Fix:**
Re-scan your system with your vulnerability scanner to ensure that ICMP timestamp messages are no longer accepted.