Security Report

Host: 10.0.2.4:8180 (tcp)

Service: www

CVE: Missing Data

CWE: Missing Data

CVSS Score: 10.0

Plugin: Apache Tomcat SEoL (<= 5.5.x) (ID: 171340)

Exploit Available: Missing Data

Version: 1.1

Exploitability Value: High

Severity: 4

Risk: Critical

Risk Value: 9.5

Description:

According to its version, Apache Tomcat is less than or equal to 5.5.x. It is, therefore, no longer maintained by its vendor or provider. Lack of support implies that no new security patches for the product will be released by the vendor. As a result, it may contain security vulnerabilities.

Possible attack scenario:

An attacker could exploit known vulnerabilities in Apache Tomcat 5.5.x to gain unauthorized access to sensitive data, disrupt business operations, or execute malicious code, potentially leading to significant financial losses and reputational damage. In a real-world scenario, an attacker might use a vulnerability to launch a denial-of-service (DoS) attack, overwhelming the server with traffic and rendering it unavailable to legitimate users, thereby impacting business continuity and customer satisfaction. The lack of security patches and support for Apache Tomcat 5.5.x also increases the risk of data breaches, intellectual property theft, and compliance violations, making it essential for organizations to upgrade to a supported version or implement alternative security measures to mitigate these risks.

Solution:

Upgrade to a version of Apache Tomcat that is currently supported.

Remediation step-by-step:

Given the information provided, the vulnerability details are not specified (N/A for Description Vulnerability, CVE, and CVSS). However, I can guide you through a general step-by-step remediation process for a vulnerability found in a service running on port 8180, which is often associated with web services (given the "www" service indicator). This process will focus on general best practices for securing a web service.

Step 1: Identify the Service and Vulnerability
- **Action**: Even though the specific vulnerability is not mentioned, identify what service is running on port 8180. Common services might include Apache Tomcat, a custom web application, or another type of web server.
- **Tools**: Use commands like `netstat -tlnp | grep 8180` (on Linux) or `netstat -an | findstr 8180` (on Windows) to identify the process ID and name of the service listening on port 8180.

Step 2: Update and Patch
- **Action**: Ensure the identified service and its dependencies are up-to-date. Even without a specific CVE, keeping software updated can mitigate many known vulnerabilities.
- **Tools**: Check the official website of the service for updates. For example, if it's Apache Tomcat, visit the Apache Tomcat download page to check for the latest version.

Step 3: Configure Secure Settings
- **Action**: Review and configure the service's settings for security best practices. This might include:
- Disabling unnecessary features or modules.
- Enabling security features like SSL/TLS for encryption.
- Setting appropriate access controls (e.g., firewall rules, access control lists).
- **Tools**: Refer to the service's documentation for configuration options. For web servers, consider using tools like SSL Labs' SSL Test to evaluate SSL/TLS configuration security.

Step 4: Implement Firewall Rules
- **Action**: Restrict access to port 8180 to only necessary sources. This could involve configuring firewall rules to limit inbound traffic.
- **Tools**: Use firewall configuration tools like `iptables` (on Linux) or Windows Defender Firewall to set up rules that restrict access to port 8180.

Step 5: Monitor for Suspicious Activity
- **Action**: Set up logging and monitoring to detect potential security issues. This includes configuring the service to log security-relevant events and regularly reviewing these logs.
- **Tools**: Utilize logging tools provided by the service or third-party solutions like ELK Stack (Elasticsearch, Logstash, Kibana) for log collection, analysis, and visualization.

Step 6: Test for Vulnerabilities
- **Action**: Perform regular vulnerability scans to identify any new or unpatched vulnerabilities in the service.
- **Tools**: Use vulnerability scanning tools like Nmap, OpenVAS, or commercial alternatives to scan the service for known vulnerabilities.

Step 7: Document and Review
- **Action**: Document all steps taken for remediation, including updates, configuration changes, and any tools or scripts used. Regularly review the security posture of the service to ensure it remains secure over time.
- **Tools**: Maintain a change management log or use version control systems like Git to track changes to configuration files and scripts.

Given the lack of specific vulnerability details, these steps focus on general security hardening and best practices for a web service running on port 8180. Always refer to the latest security guidelines and the service's official documentation for the most current and detailed advice.

Host: 10.0.2.4:8009 (tcp)

Service: ajp13

CVE: ['CVE-2020-1745', 'CVE-2020-1938']

CWE: Missing Data

CVSS Score: 9.8

Plugin: Apache Tomcat AJP Connector Request Injection (Ghostcat) (ID: 134862)

Exploit Available: true

Version: Missing Data

Exploitability Value: High

Severity: 4

Risk: High

Risk Value: 9.68

Description:

A file read/inclusion vulnerability was found in AJP connector. A remote, unauthenticated attacker could exploit this vulnerability to read web application files from a vulnerable server. In instances where the vulnerable server allows file uploads, an attacker could upload malicious JavaServer Pages (JSP) code within a variety of file types and gain remote code execution (RCE).

Possible attack scenario:

A remote attacker could exploit the AJP connector vulnerability to read sensitive configuration files, such as web.xml, and gain valuable information about the web application's architecture and security mechanisms. In a worst-case scenario, an attacker could upload malicious JSP code and achieve remote code execution, allowing them to steal sensitive data, disrupt business operations, or use the compromised server as a launching point for further attacks. The potential business risks include reputational damage, financial loss, and regulatory non-compliance, emphasizing the need for organizations to promptly patch or mitigate the vulnerability to prevent such attacks and protect their assets.

Solution:

Update the AJP configuration to require authorization and/or upgrade the Tomcat server to 7.0.100, 8.5.51, 9.0.31 or later.

Remediation step-by-step:

Remediation Steps for ajp13 Service Vulnerability

Since there is no specific CVE or CVSS score provided, we will focus on general remediation steps to secure the ajp13 service running on port 8009.

Step 1: Identify the ajp13 Service

* Determine which application or server is using the ajp13 protocol on port 8009.
* Verify that the service is necessary and not a redundant or unused configuration.

Step 2: Update and Patch

* Check for any available updates or patches for the application or server using the ajp13 protocol.

* Apply the latest security patches and updates to ensure the service is running with the latest security fixes.

Step 3: Configure Secure Settings

* Review the ajp13 configuration to ensure it is set up with secure settings, such as:
+ Disabling unnecessary features or protocols.
+ Setting up authentication and authorization mechanisms.
+ Configuring encryption (e.g., SSL/TLS) for secure communication.

Step 4: Restrict Access

* Limit access to the ajp13 service on port 8009 to only necessary IP addresses or networks.
* Use firewall rules or access control lists (ACLs) to restrict incoming connections to the service.

Step 5: Monitor and Log

* Set up monitoring and logging for the ajp13 service to detect any potential security issues or anomalies.
* Regularly review logs to identify and respond to potential security incidents.

Step 6: Consider Alternatives

* Evaluate whether the ajp13 protocol is still necessary for the application or server.
* Consider migrating to a more secure protocol, such as HTTP/2 or a modern web server protocol.

Step 7: Implement Additional Security Measures

* Consider implementing additional security measures, such as:
+ Intrusion detection and prevention systems (IDPS).
+ Web application firewalls (WAFs).
+ Regular security audits and vulnerability assessments.

By following these remediation steps, you can help secure the ajp13 service running on port 8009 and reduce the risk of potential security vulnerabilities.

Host: 10.0.2.4:6667 (tcp)

Service: irc

CVE: ['CVE-2010-2075']

CWE: Missing Data

CVSS Score: Missing Data

Plugin: UnrealIRCd Backdoor Detection (ID: 46882)

Exploit Available: true

Version: Missing Data

Exploitability Value: High

Severity: 4

Risk: Critical

Risk Value: [CVSS] Missing Data

Description:

The remote IRC server is a version of UnrealIRCd with a backdoor that allows an attacker to execute arbitrary code on the affected host.

Possible attack scenario:

A malicious actor could exploit the backdoor in the UnrealIRCd server to gain unauthorized access to the affected host, potentially leading to data breaches, lateral movement within the network, and disruption of critical services. In a real-world attack scenario, an attacker could use the compromised IRC server as a launchpad for further attacks, such as spreading malware, conducting DDoS attacks, or stealing sensitive information, ultimately causing significant financial and reputational damage to the organization. The business risks associated with this vulnerability include loss of customer trust, regulatory non-compliance, and potential legal liabilities, highlighting the need for prompt mitigation and remediation efforts to prevent or minimize the impact of a potential attack.

Solution:

Re-download the software, verify it using the published MD5 / SHA1 checksums, and re-install it.

Remediation step-by-step:

Remediation Steps for IRC Service Vulnerability

Since the specific vulnerability is not provided (N/A), we will focus on general security best practices for the IRC (Internet Relay Chat) service, which typically runs on port 6667. The goal is to secure the service and prevent potential exploitation.

Step 1: Update and Patch the IRC Server Software

1. **Identify the IRC Server Software**: Determine which IRC server software is being used (e.g., UnrealIRCd, InspIRCd).
2. **Check for Updates**: Visit the official website of the IRC server software to check for any updates or patches.
3. **Apply Updates/Patches**: Download and apply the latest updates or patches to ensure the software is up-to-date and secure.

Step 2: Configure Secure Connections

1. **Enable SSL/TLS**: Configure the IRC server to use SSL/TLS encryption for secure connections. This will protect user data and passwords from being intercepted.
2. **Generate SSL Certificates**: Generate or obtain valid SSL certificates for the IRC server.
3. **Configure Certificate Settings**: Configure the IRC server to use the generated SSL certificates.

Step 3: Implement Access Controls

1. **Set Up User Authentication**: Implement user authentication to control who can access the IRC server.
2. **Configure User Permissions**: Set up user permissions to restrict access to certain channels or features.
3. **Limit Operator Privileges**: Limit operator privileges to prevent abuse of power.

Step 4: Monitor and Log Activity

1. **Configure Logging**: Configure the IRC server to log important events, such as user connections, channel activity, and errors.
2. **Monitor Logs**: Regularly monitor logs to detect potential security issues or suspicious activity.
3. **Implement Intrusion Detection**: Consider implementing an intrusion detection system to detect and alert on potential security threats.

Step 5: Secure the Server Environment

1. **Keep the Operating System Up-to-Date**: Ensure the operating system running the IRC server is up-to-date with the latest security patches.
2. **Use a Firewall**: Configure a firewall to restrict access to the IRC server and only allow incoming connections on the necessary ports (e.g., 6667).
3. **Limit Network Exposure**: Limit the IRC server's exposure to the network by using a VPN or isolating the server from the rest of the network.

Step 6: Regularly Review and Update Security Settings

1. **Schedule Regular Security Audits**: Regularly review the IRC server's security settings and configurations to ensure they are up-to-date and secure.
2. **Update Security Settings**: Update security settings as needed to address new vulnerabilities or security concerns.
3. **Stay Informed**: Stay informed about potential security vulnerabilities and updates related to the IRC server software and operating system.

By following these remediation steps, you can help secure your IRC server and protect it from potential vulnerabilities and exploits.

Host: 10.0.2.4:5900 (tcp)

Service: vnc

CVE: Missing Data

CWE: Missing Data

CVSS Score: Missing Data

Plugin: VNC Server 'password' Password (ID: 61708)

Exploit Available: Missing Data

Version: Missing Data

Exploitability Value: High

Severity: 4

Risk: Critical

Risk Value: [CVSS] Missing Data

Description:

The VNC server running on the remote host is secured with a weak password. Nessus was able to login using VNC authentication and a password of 'password'. A remote, unauthenticated attacker could exploit this to take control of the system.

Possible attack scenario:

A weak VNC password poses a significant risk to the organization, as a remote attacker could exploit this vulnerability to gain unauthorized access to the system, potentially leading to data breaches, malware infections, or even complete system compromise. In a real-world attack scenario, an attacker could use the compromised VNC connection to move laterally within the network, escalating privileges and exploiting other vulnerabilities to maximize the impact of the breach. If exploited, this vulnerability could result in substantial business risks, including reputational damage, financial losses, and regulatory non-compliance, highlighting the need for immediate remediation, such as changing the VNC password to a strong and unique one.

Solution:

Secure the VNC service with a strong password.

Remediation step-by-step:

Remediation Steps for VNC Service Vulnerability

Service: VNC
Port: 5900

Since there is no specific CVE or CVSS score provided, we will focus on general remediation steps to secure the VNC service.

Step 1: Assess the Current Configuration (Risk Assessment)

1. Check the current VNC configuration to determine if it is using a secure protocol (e.g., TLS) for encryption.
2. Verify if authentication is required for VNC connections.
3. Identify any existing security measures, such as firewalls or access controls, that may be in place to restrict access to the VNC service.

Step 2: Secure VNC Configuration

1. **Enable Encryption**: Ensure that VNC is configured to use a secure protocol (e.g., TLS) for encryption. This will help protect data transmitted between the VNC client and server.
2. **Set up Authentication**: Configure VNC to require authentication for connections. This can include setting up username and password authentication or using alternative authentication methods like Kerberos or smart cards.
3. **Limit Access**: Restrict access to the VNC service by configuring the firewall to only allow incoming connections from trusted IP addresses or networks.

Step 3: Implement Additional Security Measures

1. **Use a VPN**: Consider requiring VNC connections to be made over a Virtual Private Network (VPN) to add an extra layer of encryption and security.
2. **Keep Software Up-to-Date**: Ensure that the VNC server and client software are updated with the latest security patches and versions.
3. **Monitor VNC Activity**: Regularly monitor VNC activity and connections to detect any potential security incidents or unauthorized access.

Step 4: Test and Verify

1. **Test VNC Connections**: Verify that VNC connections are working as expected after implementing the remediation steps.
2. **Verify Encryption**: Use tools like Wireshark or OpenSSL to verify that VNC connections are encrypted and secure.
3. **Test Authentication**: Test VNC authentication to ensure that only authorized users can access the service.

By following these remediation steps, you can help secure the VNC service and reduce the risk of potential security vulnerabilities.

Host: 10.0.2.4:5432 (tcp)

Service: postgresql

CVE: Missing Data

CWE: Missing Data

CVSS Score: 9.8

Plugin: SSL Version 2 and 3 Protocol Detection (ID: 20007)

Exploit Available: Missing Data

Version: 8.3.0 - 8.3.7

Exploitability Value: High

Severity: 4

Risk: Critical

Risk Value: 9.88

Description:

The remote service accepts connections encrypted using SSL 2.0 and/or SSL 3.0. These versions of SSL are affected by several cryptographic flaws, including: - An insecure padding scheme with CBC ciphers. - Insecure session renegotiation and resumption schemes. An attacker can exploit these flaws to conduct man-in-the-middle attacks or to decrypt communications between the affected service and clients. Although SSL/TLS has a secure means for choosing the highest supported version of the protocol (so that these versions will be used only if the client or server support nothing better), many web browsers implement this in an unsafe way that allows an attacker to downgrade a connection (such as in POODLE). Therefore, it is recommended that these protocols be disabled entirely. NIST has determined that SSL 3.0 is no longer acceptable for secure communications. As of the date of enforcement found in PCI DSS v3.1, any version of SSL will not meet the PCI SSC's definition of 'strong cryptography'.

Possible attack scenario:

A malicious actor could exploit the vulnerabilities in SSL 2.0 and SSL 3.0 to intercept sensitive information, such as credit card numbers or personal data, by conducting man-in-the-middle attacks or decrypting communications between the affected service and clients. This could lead to significant business risks, including reputational damage, financial losses, and regulatory penalties, particularly for organizations that handle sensitive information and are subject to compliance standards like PCI DSS. Furthermore, the continued use of insecure SSL versions could also lead to a loss of customer trust and potential legal liabilities, emphasizing the need for organizations to prioritize the disablement of these protocols and migrate to more secure alternatives, such as TLS.

Solution:

Consult the application's documentation to disable SSL 2.0 and 3.0. Use TLS 1.2 (with approved cipher suites) or higher instead.

Remediation step-by-step:

Remediation Steps for Postgresql Service

Since there is no specific vulnerability (CVE) mentioned, the following steps focus on general security hardening and best practices for the Postgresql service.

Step 1: Update and Patch Postgresql

1. **Check for updates**: Ensure that the Postgresql service is running with the latest version. You can check for updates using the package manager of your operating system (e.g., `apt-get` for Ubuntu/Debian or `yum` for CentOS/RHEL).
2. **Apply patches**: Apply any available security patches to the Postgresql service.

Step 2: Configure Postgresql for Secure Authentication

1. **Disable trust authentication**: Edit the `pg_hba.conf` file (usually located in `/etc/postgresql/common` or `/var/lib/postgresql/data`) and set the authentication method to `md5` or `scram-sha-256` for all connections.
2. **Set strong passwords**: Ensure that all Postgresql user accounts have strong, unique passwords.
3. **Limit login attempts**: Consider implementing a login attempt limit to prevent brute-force attacks.

Step 3: Restrict Access to Postgresql

1. **Limit listening address**: Configure Postgresql to only listen on the necessary IP address or interface. Edit the `postgresql.conf` file (usually located in `/etc/postgresql/common` or `/var/lib/postgresql/data`) and set the `listen_addresses` parameter to the desired IP address or interface.
2. **Firewall configuration**: Configure the firewall to only allow incoming connections to port 5432 from trusted IP addresses or networks.
3. **Restrict database access**: Limit access to databases and tables to only the necessary users and roles.

Step 4: Enable Logging and Monitoring

1. **Enable logging**: Configure Postgresql to log all connections, queries, and errors. Edit the `postgresql.conf` file and set the `log_destination` parameter to `stderr` or `csvlog`.
2. **Monitor logs**: Regularly monitor the Postgresql logs for suspicious activity or errors.
3. **Implement auditing**: Consider implementing an auditing tool to track all database activity.

Step 5: Regularly Back Up Data

1. **Schedule backups**: Configure a regular backup schedule for the Postgresql databases using tools like `pg_dump` or `pg_dumpall`.
2. **Store backups securely**: Store the backups in a secure location, such as an encrypted file system or a secure cloud storage service.

By following these steps, you can improve the security and integrity of your Postgresql service, even in the absence of a specific vulnerability (CVE). Remember to regularly review and update your security configurations to ensure the ongoing security of your Postgresql service.

Host: 10.0.2.4:5432 (tcp)

Service: postgresql

CVE: ['CVE-2008-0166']

CWE: 310

CVSS Score: Missing Data

Plugin: Debian OpenSSH/OpenSSL Package Random Number Generator Weakness (SSL check) (ID: 32321)

Exploit Available: true

Version: 8.3.0 - 8.3.7

Exploitability Value: High

Severity: 4

Risk: Critical

Risk Value: [CVSS] Missing Data

Description:

The remote x509 certificate on the remote SSL server has been generated on a Debian or Ubuntu system which contains a bug in the random number generator of its OpenSSL library. The problem is due to a Debian packager removing nearly all sources of entropy in the remote version of OpenSSL. An attacker can easily obtain the private part of the remote key and use this to decipher the remote session or set up a man in the middle attack.

Possible attack scenario:

In a real-world attack scenario, an attacker could exploit this vulnerability to intercept and decrypt sensitive data, such as financial information or personal identifiable information, being transmitted over the compromised SSL connection. This could lead to significant business risks, including reputational damage, financial losses, and legal liabilities, particularly for organizations handling sensitive customer data, such as e-commerce websites or online banking services. Furthermore, the vulnerability could also be used to launch targeted man-in-the-middle attacks, allowing attackers to inject malware, steal credentials, or manipulate data, ultimately compromising the integrity and trust of the affected organization's online services.

Solution:

Consider all cryptographic material generated on the remote host to be guessable. In particuliar, all SSH, SSL and OpenVPN key material should be re-generated.

Remediation step-by-step:

Remediation Steps for Postgresql Service

Since there is no specific vulnerability (CVE) mentioned, the following steps will focus on general security best practices for the Postgresql service.

Step 1: Update and Patch Postgresql

1. Check the current version of Postgresql: `postgres -V` or `psql -V`
2. Visit the official Postgresql website to check for the latest version.
3. Update Postgresql to the latest version using the package manager (e.g., `apt-get` or `yum`) or by downloading and installing the latest version from the official website.
4. Apply any available patches to ensure the service is up-to-date.

Step 2: Secure Postgresql Configuration

1. Edit the `postgresql.conf` file (usually located at `/etc/postgresql/common/postgresql.conf` or `/var/lib/postgresql/data/postgresql.conf`).
2. Set `listen_addresses` to a specific IP address or localhost (`127.0.0.1`) to restrict access to the service.
3. Set `ssl` to `on` to enable SSL/TLS encryption.
4. Set `ssl_cert_file` and `ssl_key_file` to the paths of the SSL certificate and private key files.
5. Restart the Postgresql service to apply the changes.

Step 3: Configure Authentication and Authorization

1. Edit the `pg_hba.conf` file (usually located at `/etc/postgresql/common/pg_hba.conf` or `/var/lib/postgresql/data/pg_hba.conf`).
2. Set up authentication methods (e.g., `md5` or `scram-sha-256`) for each user and database.
3. Configure authorization settings (e.g., `host`, `hostssl`, or `local`) to restrict access to the service.
4. Restart the Postgresql service to apply the changes.

Step 4: Limit Network Exposure

1. Use a firewall to restrict access to the Postgresql port (5432) to only trusted IP addresses or networks.
2. Consider using a VPN or SSH tunnel to encrypt traffic to and from the Postgresql service.

Step 5: Monitor and Audit Postgresql

1. Set up logging and monitoring tools (e.g., `pg_badger` or `postgres-log- analyzer`) to track Postgresql activity.
2. Regularly review logs and audit trails to detect potential security issues.
3. Consider implementing a Web Application Firewall (WAF) to protect against common web attacks.

Step 6: Regularly Back Up Data

1. Set up regular backups of Postgresql databases using tools like `pg_dump` or `pg_dumpall`.
2. Store backups securely, ideally in an encrypted and version-controlled manner.

By following these steps, you can improve the security posture of your Postgresql service and reduce the risk of potential vulnerabilities. Regularly review and update your security configurations to ensure the service remains secure.

Host: 10.0.2.4:1524 (tcp)

Service: wild_shell

CVE: Missing Data

CWE: Missing Data

CVSS Score: 9.8

Plugin: Bind Shell Backdoor Detection (ID: 51988)

Exploit Available: Missing Data

Version: Missing Data

Exploitability Value: High

Severity: 4

Risk: Critical

Risk Value: 9.38

Description:

A shell is listening on the remote port without any authentication being required. An attacker may use it by connecting to the remote port and sending commands directly.

Possible attack scenario:

In a real-world attack scenario, an unauthorized user could exploit the unauthenticated shell to gain remote access to the system, allowing them to steal sensitive data, disrupt operations, or install malware. This vulnerability poses significant business risks, including data breaches, intellectual property theft, and reputational damage, which can result in financial losses and regulatory penalties. Furthermore, an attacker could use the compromised system as a launching point for lateral movement, potentially compromising other systems and networks, and amplifying the overall impact of the breach on the organization's security and operations.

Solution:

Verify if the remote host has been compromised, and reinstall the system if necessary.

Remediation step-by-step:

Remediation Steps for Wild Shell Service

Since the vulnerability description, CVE, and CVSS score are not available, we will focus on general remediation steps to secure the Wild Shell service running on port 1524.

Step 1: Identify and Assess the Service

* Verify the Wild Shell service is necessary for your system's operation.
* Assess the potential risks associated with the service, considering its functionality and exposure.

Step 2: Update and Patch the Service

* Check the official website or repository of the Wild Shell service for any available updates or patches.
* Apply the latest updates or patches to ensure you have the most secure version of the service.

Step 3: Configure the Service Securely

* Review the service's configuration files and settings to ensure they are set to the most secure options.
* Disable any unnecessary features or plugins that may increase the attack surface.

Step 4: Restrict Access to the Service

* Limit access to the Wild Shell service by configuring the firewall to only allow incoming connections from trusted IP addresses or networks.
* Consider using authentication and authorization mechanisms to restrict access to authorized users.

Step 5: Monitor the Service

* Regularly monitor the Wild Shell service for suspicious activity, such as unusual login attempts or unexpected changes to the service's configuration.
* Set up logging and alerting mechanisms to notify you of potential security incidents.

Step 6: Consider Alternative Services

* If the Wild Shell service is not essential, consider replacing it with a more secure alternative.
* Evaluate the security features and reputation of potential replacement services.

Step 7: Implement Additional Security Measures

* Consider implementing additional security measures, such as:
+ Intrusion Detection and Prevention Systems (IDPS)
+ Web Application Firewalls (WAF)

+ Regular security audits and penetration testing

By following these steps, you can help reduce the risk associated with the Wild Shell service and improve the overall security of your system.

Host: 10.0.2.4:80 (tcp)

Service: www

CVE: Missing Data

CWE: Missing Data

CVSS Score: 10.0

Plugin: Canonical Ubuntu Linux SEoL (8.04.x) (ID: 201352)

Exploit Available: Missing Data

Version: 2.2.8

Exploitability Value: High

Severity: 4

Risk: Critical

Risk Value: 9.5

Description:

According to its version, Canonical Ubuntu Linux is 8.04.x. It is, therefore, no longer maintained by its vendor or provider. Lack of support implies that no new security patches for the product will be released by the vendor. As a result, it may contain security vulnerabilities.

Possible attack scenario:

An attacker could exploit known vulnerabilities in Ubuntu 8.04.x to gain unauthorized access to sensitive data or disrupt business operations, potentially leading to financial losses and reputational damage. The lack of security patches also increases the risk of malware infections, ransomware attacks, and other types of cyber threats, which could compromise business continuity and impact customer trust. If an attack occurs, the organization may face significant business risks, including regulatory non-compliance, intellectual property theft, and costly remediation efforts, highlighting the need for urgent upgrade or migration to a supported version of the operating system.

Solution:

Upgrade to a version of Canonical Ubuntu Linux that is currently supported.

Remediation step-by-step:

Remediation Steps for Vulnerability on Port 80 (www service)

Since the specific vulnerability details (CVE, CVSS, etc.) are not provided, the following steps are general recommendations to enhance the security of the www service running on port 80. These steps focus on best practices for securing web servers and may not directly address the unspecified vulnerability.

1. **Update and Patch the Web Server Software**

- **Action**: Ensure the web server software (e.g., Apache, Nginx, IIS) and its components are up-to-date with the latest security patches.
- **Rationale**: Outdated software can leave the server vulnerable to known exploits.
- **Steps**:
1. Identify the web server software in use.
2. Check for updates from the official software repository.
3. Apply the latest security patches and updates.

2. **Configure Secure Communication (HTTPS)**

- **Action**: Enable HTTPS (SSL/TLS) for the website to encrypt data in transit.
- **Rationale**: Using HTTPS protects user data and is now considered a best practice for all websites.
- **Steps**:
1. Obtain an SSL/TLS certificate from a trusted Certificate Authority (CA) or use a free service like Let's Encrypt.
2. Configure the web server to use the SSL/TLS certificate for HTTPS connections.
3. Redirect all HTTP traffic to HTTPS.

3. **Implement Security Headers**

- **Action**: Configure the web server to include security headers in its HTTP responses.
- **Rationale**: Security headers can help protect against various types of attacks, including XSS and clickjacking.
- **Steps**:

1. Research and decide on the appropriate security headers to implement (e.g., Content-Security-Policy, X-Frame-Options, Strict-Transport-Security).
2. Configure the web server to include these headers in its responses.

4. **Limit Information Disclosure**

- **Action**: Minimize the information disclosed by the web server about its configuration and software.
- **Rationale**: Reducing information disclosure makes it harder for attackers to identify potential vulnerabilities.
- **Steps**:
1. Configure the web server to hide version numbers and other identifying information in error messages and headers.
2. Remove any unnecessary or sensitive files from the web root directory.

5. **Monitor for Suspicious Activity**

- **Action**: Set up logging and monitoring to detect potential security incidents.
- **Rationale**: Early detection of suspicious activity can help mitigate the impact of a security breach.
- **Steps**:
1. Ensure comprehensive logging is enabled on the web server.
2. Implement a monitoring solution to analyze logs for signs of suspicious activity.
3. Set up alerts for potential security incidents.

6. **Regularly Audit and Test Security**

- **Action**: Perform regular security audits and penetration testing.
- **Rationale**: Regular audits and testing help identify vulnerabilities before they can be exploited.
- **Steps**:
1. Schedule regular security audits to review configurations and identify potential vulnerabilities.
2. Conduct penetration testing to simulate attacks and assess the web server's defenses.

By following these steps, you can significantly enhance the security posture of your www service running on port 80, even without specific details on the vulnerability. Remember, security is an ongoing process, and regular monitoring and updates are crucial to maintaining a secure environment.

Host: 10.0.2.4:25 (tcp)

Service: smtp

CVE: Missing Data

CWE: Missing Data

CVSS Score: 9.8

Plugin: SSL Version 2 and 3 Protocol Detection (ID: 20007)

Exploit Available: Missing Data

Version: Missing Data

Exploitability Value: High

Severity: 4

Risk: Critical

Risk Value: 9.68

Description:

The remote service accepts connections encrypted using SSL 2.0 and/or SSL 3.0. These versions of SSL are affected by several cryptographic flaws, including: - An insecure padding scheme with CBC ciphers. - Insecure session renegotiation and resumption schemes. An attacker can exploit these flaws to conduct man-in-the-middle attacks or to decrypt communications between the affected service and clients. Although SSL/TLS has a secure means for choosing the highest supported version of the protocol (so that these versions will be used only if the client or server support nothing better), many web browsers implement this in an unsafe way that allows an attacker to downgrade a connection (such as in POODLE). Therefore, it is recommended that these protocols be disabled entirely. NIST has determined that SSL 3.0 is no longer acceptable for secure communications. As of the date of enforcement found in PCI DSS v3.1, any version of SSL will not meet the PCI SSC's definition of 'strong cryptography'.

Possible attack scenario:

A malicious actor could exploit the vulnerabilities in SSL 2.0 and 3.0 to intercept sensitive data, such as credit card numbers or personal identifiable information, by conducting man-in-the-middle attacks or decrypting communications between the service and clients. This could lead to significant business risks, including reputational damage, financial losses, and regulatory non-compliance, particularly for organizations subject to PCI DSS standards. Furthermore, the continued use of insecure SSL protocols could also expose businesses to potential lawsuits and fines, as well as loss of customer trust, ultimately jeopardizing their overall operations and bottom line.

Solution:

Consult the application's documentation to disable SSL 2.0 and 3.0. Use TLS 1.2 (with approved cipher suites) or higher instead.

Remediation step-by-step:

Remediation Steps for SMTP Service Vulnerability

Although a specific vulnerability is not identified, it's essential to follow best practices to secure the SMTP service. Here's a step-by-step remediation plan:

Step 1: Configure SMTP Server to Use Secure Connections

1. Enable Transport Layer Security (TLS) or Secure Sockets Layer (SSL) encryption on the SMTP server.
2. Ensure that the SMTP server is configured to use a secure port (e.g., 465 for SSL or 587 for TLS).
3. Update the SMTP server configuration to require encryption for all incoming and outgoing connections.

Step 2: Implement Authentication and Authorization

1. Configure the SMTP server to require authentication for all users.
2. Implement a secure authentication mechanism, such as SASL (Simple Authentication and Security Layer) or SMTP AUTH.
3. Ensure that only authorized users have access to the SMTP server.

Step 3: Restrict Access to the SMTP Server

1. Configure the SMTP server to only listen on specific IP addresses or interfaces.
2. Restrict access to the SMTP server using firewall rules or access control lists (ACLs).
3. Limit the number of concurrent connections to the SMTP server to prevent abuse.

Step 4: Update and Patch the SMTP Server

1. Ensure that the SMTP server software is up-to-date with the latest security patches.
2. Regularly review and apply security updates to the SMTP server software.
3. Consider implementing a vulnerability management program to identify and remediate potential vulnerabilities.

Step 5: Monitor and Log SMTP Server Activity

1. Configure the SMTP server to log all connections, authentication attempts, and mail transactions.
2. Monitor the SMTP server logs regularly to detect potential security issues.
3. Implement an intrusion detection system (IDS) or a security information and event management (SIEM) system to monitor SMTP server activity.

Step 6: Implement Anti-Spam and Anti-Virus Measures

1. Configure the SMTP server to use anti-spam and anti-virus software.
2. Implement a content filtering system to block malicious emails.
3. Regularly update the anti-spam and anti-virus software to ensure protection against the latest threats.

By following these remediation steps, you can help secure your SMTP server and reduce the risk of potential vulnerabilities. Regularly review and update your security measures to ensure the continued security of your SMTP server.

Host: 10.0.2.4:25 (tcp)

Service: smtp

CVE: ['CVE-2008-0166']

CWE: 310

CVSS Score: Missing Data

Plugin: Debian OpenSSH/OpenSSL Package Random Number Generator Weakness (SSL check) (ID: 32321)

Exploit Available: true

Version: Missing Data

Exploitability Value: High

Severity: 4

Risk: Critical

Risk Value: [CVSS] Missing Data

Description:

The remote x509 certificate on the remote SSL server has been generated on a Debian or Ubuntu system which contains a bug in the random number generator of its OpenSSL library. The problem is due to a Debian packager removing nearly all sources of entropy in the remote version of OpenSSL. An attacker can easily obtain the private part of the remote key and use this to decipher the remote session or set up a man in the middle attack.

Possible attack scenario:

In a real-world attack scenario, an attacker could exploit the vulnerable x509 certificate to intercept and decrypt sensitive data, such as financial information or personal identifiable information, transmitted between the client and server. This could lead to significant business risks, including reputational damage, financial loss, and legal liability, particularly for organizations that handle sensitive customer data, such as e-commerce companies or financial institutions. Furthermore, a successful man-in-the-middle attack could

also enable an attacker to inject malware or steal sensitive information, potentially compromising the entire network and causing long-term damage to the organization's security posture and customer trust.

Solution:

Consider all cryptographic material generated on the remote host to be guessable. In particuliar, all SSH, SSL and OpenVPN key material should be re-generated.

Remediation step-by-step:

Remediation Steps for SMTP Service Vulnerability

Although no specific vulnerability is mentioned, we can still provide general remediation steps to secure the SMTP service running on port 25.

Step 1: Update and Patch the SMTP Server

1. Check the version of the SMTP server software (e.g., Postfix, Sendmail, or Microsoft Exchange).
2. Ensure the SMTP server is updated to the latest version.
3. Apply any available security patches to fix known vulnerabilities.

Step 2: Configure SMTP Server Settings

1. **Authentication**: Enable authentication mechanisms like SMTP AUTH (e.g., PLAIN, LOGIN, or CRAM-MD5) to prevent unauthorized access.
2. **Encryption**: Configure the SMTP server to use Transport Layer Security (TLS) or Secure Sockets Layer (SSL) encryption to protect data in transit.
3. **Access Control**: Restrict access to the SMTP server by configuring IP address-based access control lists (ACLs) or firewall rules.

Step 3: Implement Security Best Practices

1. **Use strong passwords**: Ensure all user accounts, including administrative accounts, have strong, unique passwords.
2. **Limit relay access**: Restrict relay access to authorized IP addresses or networks to prevent spamming and unauthorized mail relaying.
3. **Monitor logs**: Regularly monitor SMTP server logs to detect potential security issues or suspicious activity.

Step 4: Configure Firewall Rules

1. **Restrict incoming traffic**: Configure the firewall to only allow incoming traffic on port 25 from trusted IP addresses or networks.
2. **Block outgoing traffic**: Block outgoing traffic on port 25 to prevent spamming or unauthorized mail relaying.

Step 5: Regularly Scan for Vulnerabilities

1. **Use vulnerability scanners**: Utilize tools like OpenVAS, Nessus, or Qualys to regularly scan the SMTP server for potential vulnerabilities.
2. **Address identified vulnerabilities**: Remediate any identified vulnerabilities according to the scanner's recommendations.

By following these remediation steps, you can help secure your SMTP service and reduce the risk of potential vulnerabilities.

Host: 10.0.2.4:22 (tcp)

Service: ssh

CVE: ['CVE-2008-0166']

CWE: 310

CVSS Score: Missing Data

Plugin: Debian OpenSSH/OpenSSL Package Random Number Generator Weakness (ID: 32314)

Exploit Available: true

Version: 4.7p1 Debian 8ubuntu1

Exploitability Value: High

Severity: 4

Risk: Critical

Risk Value: [CVSS] Missing Data

Description:

The remote SSH host key has been generated on a Debian or Ubuntu system which contains a bug in the random number generator of its OpenSSL library. The problem is due to a Debian packager removing nearly all sources of entropy in the remote version of OpenSSL. An attacker can easily obtain the private part of the remote key and use this to set up decipher the remote session or set up a man in the middle attack.

Possible attack scenario:

In a real-world attack scenario, an attacker could exploit the weakened SSH host key to intercept and decrypt sensitive data transmitted between the remote host and clients, potentially leading to unauthorized access to confidential information. This vulnerability could also enable man-in-the-middle attacks, allowing an attacker to impersonate the remote host and trick clients into revealing sensitive credentials or data, posing a significant risk to business operations and reputation. If exploited, this vulnerability could result in significant business risks, including data breaches, financial losses, and regulatory non-compliance, highlighting the need for urgent remediation and key regeneration to prevent such attacks.

Solution:

Consider all cryptographic material generated on the remote host to be guessable. In particuliar, all SSH, SSL and OpenVPN key material should be re-generated.

Remediation step-by-step:

Remediation Steps for SSH Service Vulnerability

Although no specific vulnerability is mentioned, it's essential to follow best practices to secure the SSH service. Here's a step-by-step remediation plan:

Step 1: Update and Patch SSH Server

1. Check the current version of the SSH server: `ssh -V` (for OpenSSH) or check the package manager for the installed version.
2. Update the SSH server to the latest version using the package manager (e.g., `apt-get update && apt-get install openssh-server` for Ubuntu-based systems).
3. Apply any available patches for the SSH server.

Step 2: Configure SSH Server

1. Edit the SSH server configuration file (usually `/etc/ssh/sshd_config`).
2. Set the following parameters:
* `Port 22` (or a non-standard port, if desired, for added security).
* `Protocol 2` (to disable SSH protocol 1, which is insecure).
* `PermitRootLogin no` (to prevent root login via SSH).
* `MaxAuthTries 3` (to limit the number of authentication attempts).
* `AllowUsers` or `AllowGroups` to restrict access to specific users or groups.
3. Restart the SSH server service: `service ssh restart` (or `systemctl restart ssh` for systemd-based systems).

Step 3: Implement Strong Authentication and Authorization

1. Enable public key authentication:
* Generate a public-private key pair using a tool like `ssh-keygen`.
* Copy the public key to the server (e.g., `ssh-copy-id user@hostname`).
2. Disable password authentication:
* Set `PasswordAuthentication no` in the SSH server configuration file.
3. Consider implementing two-factor authentication (2FA) or multi-factor authentication (MFA) for added security.

Step 4: Limit Access and Monitor Activity

1. Restrict access to the SSH server:
* Use a firewall to limit incoming connections to the SSH port (e.g., `ufw allow ssh` for Ubuntu-based systems).
* Configure IP filtering to only allow connections from trusted IP addresses.
2. Monitor SSH server activity:
* Regularly review SSH server logs (e.g., `/var/log/auth.log` for Ubuntu-based systems).
* Consider using a log analysis tool or a security information and event management (SIEM) system.

Step 5: Regularly Review and Update Configuration

1. Schedule regular reviews of the SSH server configuration and logs.
2. Update the configuration as needed to ensure the SSH server remains secure and compliant with organizational security policies.

By following these remediation steps, you can help ensure the security and integrity of your SSH service, even in the absence of a specific vulnerability.