Class 9 - 04/09/2023

Report Procedure Structure:

Vulnerability name

CWE it belongs to

OSWAP it belongs to

Description

Business Impact

Affected url

POC(Proof of concept)

Remediation

Nessus is used to scan and get information about the vulnerabilities and everything else unlike nmap.

shodan.io

To calculate CVSS Score: https://www.first.org/cvss/calculator/3.0

Task - Vulnerability Report

We are scanning Christ University official website for vulnerabilities.

Url: https://christuniversity.in/

The result from the scan:

111.93.136.229

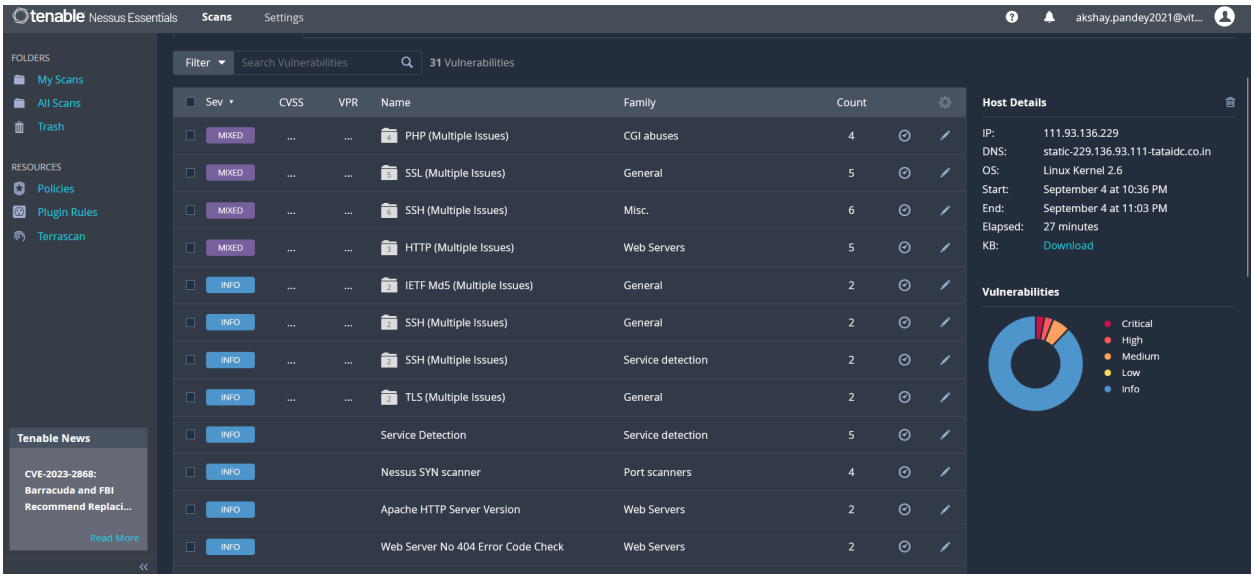
CRITICAL	HIGH	MEDIUM	LOW	INFO
2	2	3	3	50

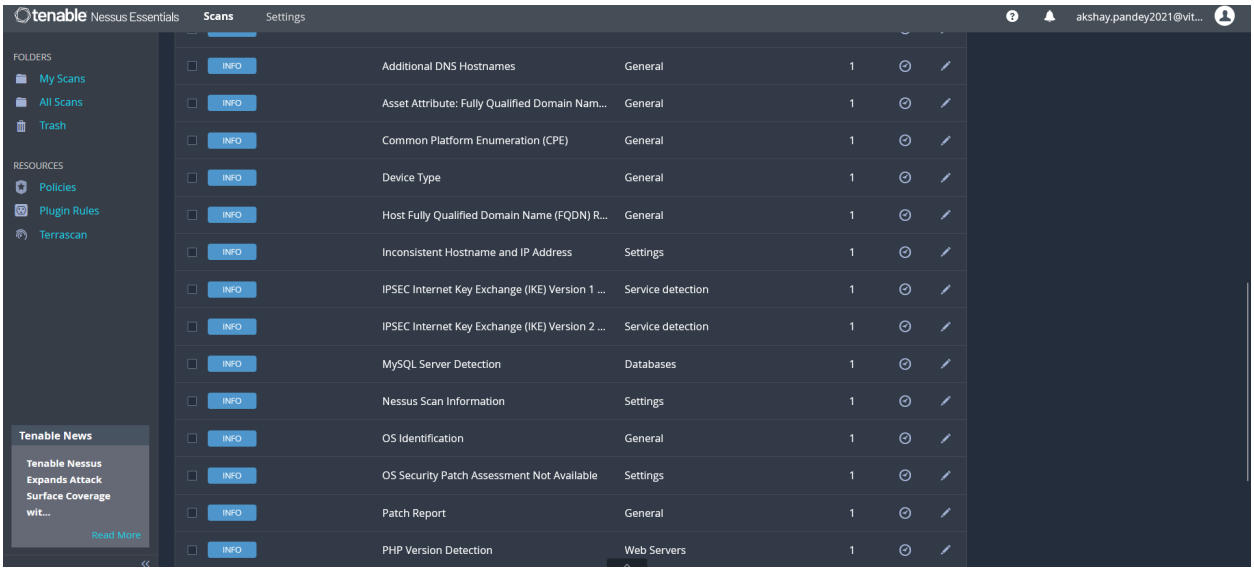
Scan Information

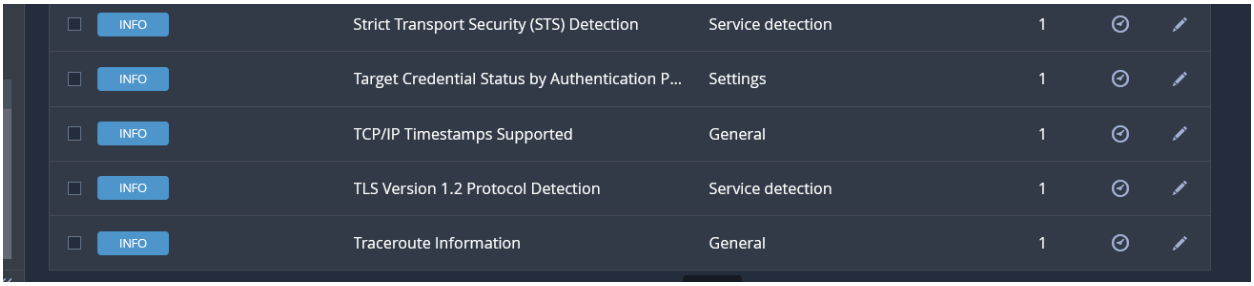
Start time:	Mon Sep 4 22:36:46 2023
End time:	Mon Sep 4 23:03:45 2023

Host Information

DNS Name:	static-229.136.93.111-tataidc.co.in
IP:	111.93.136.229
OS:	Linux Kernel 2.6







In the report we will only give report on vulnerability that are critical, high and medium because they are more vulnerable to attacks.

Vulnerabilities Reports:

1. **130276 - PHP < 7.1.33 / 7.2.x < 7.2.24 / 7.3.x < 7.3.11 Remote Code Execution Vulnerability. → Critical**

Vulnerability Name: PHP Remote Code Execution Vulnerability (CVE-2019-11043)

CWE: CWE-20: Improper Input Validation

OWASP: This vulnerability may be related to OWASP Top Ten category A3: Sensitive Data Exposure, as it can lead to unauthorized access and execution of arbitrary code.

Description:
The PHP version running on the remote web server is vulnerable to a remote code execution attack due to inadequate validation of user input. Specifically, this vulnerability exists in PHP versions prior to 7.1.33, 7.2.x prior to 7.2.24, and 7.3.x prior to 7.3.11. An

attacker, without authentication, can exploit this vulnerability by sending a specially crafted request that breaks the `fastcgi_split_path_info` directive, allowing them to execute arbitrary code on the target server.

Business Impact:
This vulnerability poses a high risk to the business as it can lead to unauthorized access and execution of arbitrary code on the affected web server. An attacker could potentially compromise the confidentiality, integrity, and availability of data and services hosted on the server. This could result in data breaches, data loss, and service disruption, leading to reputational damage and potential legal consequences.

Affected URL:https://111.93.136.229/ (Running PHP version 5.6.40 under X-Powered-By: PHP/5.6.40)

Proof of Concept (PoC):
Exploiting this vulnerability can be done using a specially crafted HTTP request. However, providing a PoC in this context is not appropriate, as it would involve demonstrating a real-world attack, which is illegal and unethical. It is strongly advised not to attempt any PoC without proper authorization.

Remediation:
To mitigate this vulnerability, it is recommended to follow these steps:

- **Upgrade PHP:** Upgrade the PHP version to at least 7.3.11 or later, as recommended by the vendor.
- **Update Web Server:** Ensure that the web server and associated components are also up-to-date and properly configured.
- **Review Code:** Review the web application code for any vulnerabilities and sanitize user inputs to prevent similar issues in the future.
- **Security Patching:** Regularly apply security patches and updates to the server and its software components.
- **Access Control:** Implement proper access controls and firewall rules to restrict unauthorized access to the server.
- **Monitoring:** Set up monitoring and intrusion detection systems to detect and respond to any suspicious activity.

- **Backup:** Regularly back up critical data and configurations to facilitate recovery in case of a successful attack.
- **Incident Response Plan:** Develop and test an incident response plan to handle security incidents effectively.

2. **58987 - PHP Unsupported Version Detection → Critical**

Vulnerability Name: Unsupported PHP Version Detection

CWE: CWE-732: Incorrect Permission Assignment for Critical Resource

OWASP: This vulnerability is not directly related to the OWASP Top Ten categories but is a security best practice issue.

Description:
The remote host contains an unsupported version of PHP, specifically version 5.6.40, which is no longer supported by the PHP community. Lack of support implies that no new security patches will be released by the vendor. This makes the system likely to contain security vulnerabilities that can be exploited by attackers. The end of support for PHP 5.6.40 was on December 31, 2018, and it is important to note that unsupported software poses a significant security risk.

Business Impact:
The use of an unsupported PHP version poses a critical risk to the business. Unsupported software may contain known vulnerabilities for which no security patches are available. These vulnerabilities can be exploited by attackers to gain unauthorized access to the server, compromise data, disrupt services, and potentially lead to data breaches or service outages. Additionally, it can result in non-compliance with industry and regulatory standards.

Affected URL:https://111.93.136.229/ (Running PHP version 5.6.40 under X-Powered-By: PHP/5.6.40)

Proof of Concept (PoC):
This vulnerability does not have a direct PoC, as it is more about the lack of support and potential security vulnerabilities associated with unsupported software. Exploiting this vulnerability involves searching for known vulnerabilities in PHP 5.6.40 and attempting to exploit them, which is not advisable.

Remediation:
To mitigate this vulnerability and ensure the security of the system, follow these steps:

- **Upgrade PHP:** Upgrade PHP to a currently supported version, such as PHP 8.0.x or 8.1.x, as recommended by the PHP community.
- **Regularly Update:** Ensure that PHP and other software components are regularly updated to stay current with security patches.
- **Monitoring:** Implement monitoring systems to detect and respond to security incidents promptly.
- **Security Best Practices:** Follow security best practices for web application development, server configuration, and access control.
- **Backup:** Regularly back up critical data and configurations to facilitate recovery in case of security incidents.
- **Compliance:** Ensure that the server and software are in compliance with industry and regulatory standards.
- **Risk Assessment:** Perform regular risk assessments to identify and address security weaknesses.
- **Retire Legacy Software:** Consider migrating or retiring legacy software versions that are no longer supported.

3. **142591 - PHP < 7.3.24 Multiple Vulnerabilities → High**

Vulnerability Name: PHP < 7.3.24 Multiple Vulnerabilities

CWE: This vulnerability may be associated with various Common Weakness Enumeration (CWE) identifiers, depending on the specific vulnerabilities addressed in PHP 7.3.24.

OWASP: This vulnerability may be related to OWASP Top Ten category A9: Using Components with Known Vulnerabilities.

Description:
The version of PHP running on the remote web server is reported as being prior to 7.3.24. This outdated PHP version is affected by multiple vulnerabilities. Unfortunately, the specific vulnerabilities are not detailed in the provided information, but it is essential

to understand that running an older version of PHP can expose the system to various security risks.

Business Impact:
The use of an outdated PHP version with known vulnerabilities poses a medium risk to the business. These vulnerabilities could potentially be exploited by attackers to compromise the web server, disrupt services, steal data, or perform other malicious actions. The impact may vary depending on the specific vulnerabilities present and the nature of the web application.

Affected URL:
The affected URL running PHP version 5.6.40 is https://111.93.136.229/ (identified under X-Powered-By: PHP/5.6.40).

Proof of Concept (PoC):
A Proof of Concept (PoC) for these unspecified vulnerabilities is not provided in the given information. Exploiting PHP vulnerabilities typically requires detailed knowledge of the specific flaws and is not advisable without proper authorization.

Remediation:
To mitigate these vulnerabilities and ensure the security of the web server, follow these steps:

- **Upgrade PHP:** Upgrade PHP to version 7.3.24 or a later version, as recommended by the PHP community.
- **Regularly Update:** Ensure that PHP and other software components are regularly updated to stay current with security patches.
- **Monitoring:** Implement monitoring systems to detect and respond to security incidents promptly.
- **Security Best Practices:** Follow security best practices for web application development, server configuration, and access control.
- **Backup:** Regularly back up critical data and configurations to facilitate recovery in case of security incidents.
- **Vulnerability Assessment:** Perform regular vulnerability assessments and security scans to identify and address vulnerabilities in the web application and server.

- **Risk Assessment:** Conduct a risk assessment to evaluate the potential impact of these vulnerabilities on the organization and prioritize remediation efforts accordingly.

4. 42873 - SSL Medium Strength Cipher Suites Supported (SWEET32) → High

Vulnerability Name: SSL Medium Strength Cipher Suites Supported (SWEET32)

CWE: CWE-326: Inadequate Encryption Strength

OWASP: This vulnerability is not directly related to the OWASP Top Ten categories but is a security best practice issue.

Description:
The remote service (likely a web server) supports the use of SSL ciphers that provide medium-strength encryption. Medium-strength encryption is defined as encryption that uses key lengths of at least 64 bits and less than 112 bits or uses the 3DES (Triple Data Encryption Standard) encryption suite. These medium-strength ciphers are considered less secure, and their use can potentially expose the system to security risks.

Business Impact:
The use of medium-strength SSL ciphers can pose a medium risk to the business. While medium-strength encryption is still relatively secure, it is more susceptible to attacks compared to stronger encryption methods. Attackers on the same physical network might find it easier to circumvent medium-strength encryption, potentially compromising the confidentiality and integrity of data transmitted over SSL/TLS connections.

Affected URL:
The affected URL running SSL/TLS with medium-strength ciphers is not explicitly mentioned in the provided information but can be assumed to be the web server's main address.

Proof of Concept (PoC):
A Proof of Concept (PoC) is not provided in the given information, as this vulnerability is more about the configuration of SSL/TLS ciphers rather than a specific exploit.

Remediation:
To mitigate this vulnerability and enhance the security of SSL/TLS communications, follow these steps:

- **Update SSL/TLS Configuration:** Reconfigure the affected application or web server to avoid the use of medium-strength ciphers.
- **Use Stronger Ciphers:** Configure the SSL/TLS settings to use strong encryption ciphers with key lengths of at least 128 bits, such as AES (Advanced Encryption Standard) ciphers.
- **Remove 3DES:** If the server is using 3DES, consider removing it from the list of supported ciphers, as it is considered weak and no longer recommended.
- **SSL/TLS Hardening:** Implement SSL/TLS hardening practices to ensure that the server uses modern and secure cryptographic algorithms and configurations.
- **Regular Updates:** Keep the server's SSL/TLS libraries and software up-to-date to benefit from security improvements and patches.
- **Network Segmentation:** Implement network segmentation to limit exposure to potential attackers on the same physical network.
- **Monitoring:** Set up monitoring and intrusion detection systems to detect and respond to any suspicious SSL/TLS activity.
- **Compliance:** Ensure that SSL/TLS configurations comply with industry standards and regulatory requirements.

5. **11213 - HTTP TRACE / TRACK Methods Allowed → Medium**

Vulnerability Name: HTTP TRACE / TRACK Methods Allowed

CWE: CWE-16: Configuration

OWASP: This vulnerability is not directly related to the OWASP Top Ten categories but is a security best practice issue.

Description:
The remote web server supports the TRACE and/or TRACK methods. TRACE and TRACK are HTTP methods that are typically used for debugging web server connections. Enabling these methods in a production environment can pose security risks, as they may leak sensitive information and expose the system to potential vulnerabilities.

Business Impact:
Allowing the use of HTTP TRACE and TRACK methods in a production environment poses a medium risk to the business. Attackers can potentially abuse these methods to gather information about the web server, potentially leading to security vulnerabilities or exposure of sensitive data. It is considered a security best practice to disable these methods in production to reduce exposure to potential threats.

Affected URL:
The affected URL where HTTP TRACE and TRACK methods are allowed is not explicitly mentioned in the provided information, but it is likely to be the main web server address.

Proof of Concept (PoC):
A Proof of Concept (PoC) is not provided in the given information, as this vulnerability is more about the server's configuration than a specific exploit.

Remediation:
To mitigate this vulnerability and enhance the security of the web server, follow these steps:

- **Disable TRACE and TRACK Methods:** Disable the TRACE and TRACK methods in the web server configuration. This can typically be done by adding the following lines to the server's configuration file for each virtual host:

  ```
  RewriteEngine on
  RewriteCond %{REQUEST_METHOD} ^(TRACE|TRACK)
  RewriteRule .* - [F]
  ```

  Alternatively, note that Apache versions 1.3.34, 2.0.55, and 2.2 support disabling the TRACE method natively via the 'TraceEnable' directive.

- **Regularly Update:** Ensure that the web server software is regularly updated to the latest version to benefit from security improvements and patches.
- **Security Best Practices:** Follow security best practices for web server configuration, access control, and permissions.
- **Monitoring:** Set up monitoring and intrusion detection systems to detect and respond to any suspicious activity related to HTTP TRACE and TRACK methods.
- **Compliance:** Ensure that the web server configuration complies with industry standards and regulatory requirements.

6. **152853 - PHP < 7.3.28 Email Header Injection → Medium**

Vulnerability Name: PHP < 7.3.28 Email Header Injection

CWE: CWE-93: Improper Neutralization of CRLF Sequences in HTTP Headers ('HTTP Response Splitting')

OWASP: This vulnerability is not directly related to the OWASP Top Ten categories but relates to web application security.

Description:
The version of PHP running on the remote web server is reported as prior to 7.3.28. This PHP version is affected by an email header injection vulnerability due to a failure to properly handle CRLF (Carriage Return Line Feed) sequences in email header fields. An unauthenticated remote attacker can exploit this vulnerability by inserting line feed characters into email headers, potentially gaining full control over the email header content.

Business Impact:
The email header injection vulnerability poses a medium risk to the business. If exploited, an attacker could manipulate email headers, potentially leading to various malicious actions, including phishing, email spoofing, or altering email content. Such attacks can undermine the integrity and trustworthiness of email communication, which can have serious implications for business operations and reputation.

Affected URL:
The affected URL running PHP version 5.6.40 with the email header injection vulnerability is identified as https://111.93.136.229/.

Proof of Concept (PoC):
A Proof of Concept (PoC) is not provided in the given information. However, the vulnerability description outlines the possibility of an attacker injecting line feed characters into email headers. Developing a specific PoC would require knowledge of the target application's email handling code, which is not provided in the plugin output.

Remediation:
To mitigate this vulnerability and enhance the security of the PHP application, follow these steps:

- **Upgrade PHP:** Upgrade the PHP version to 7.3.28 or later. This version includes fixes for the email header injection vulnerability.
- **Security Patching:** Ensure that PHP and other software components are regularly updated with security patches to address known vulnerabilities.
- **Input Validation:** Implement strict input validation and sanitization when processing user-supplied data that is used in email headers. Validate and sanitize user inputs to prevent the injection of malicious characters.
- **Web Application Firewall (WAF):** Implement a Web Application Firewall to detect and block potentially malicious requests, including those attempting email header injection.
- **Email Content Filtering:** Use email content filtering mechanisms to detect and block suspicious or malicious emails, which may help mitigate the impact of email header injection attacks.
- **Secure Coding Practices:** Train developers on secure coding practices, including proper handling of user inputs and email-related functions to prevent email header injection vulnerabilities.
- **Security Testing:** Regularly perform security testing, including vulnerability scanning and penetration testing, to identify and address security issues proactively.

7. **90317 - SSH Weak Algorithms Supported → Medium**

Vulnerability Name: SSH Weak Algorithms Supported

CWE: CWE-326: Inadequate Encryption Strength

OWASP: This vulnerability is not directly related to the OWASP Top Ten categories but is a security best practice issue.

Description:
The remote SSH (Secure Shell) server is configured to allow weak encryption algorithms or no encryption algorithm at all. Specifically, it supports the Arcfour stream cipher, which is considered weak due to vulnerabilities with weak keys. Additionally, there is support for Arcfour128 and Arcfour256, which are variations of Arcfour with different key lengths.

Business Impact:
The support for weak encryption algorithms in the SSH server poses a medium risk to the business. Weak encryption algorithms can potentially be exploited by attackers to compromise the confidentiality and integrity of data transmitted over SSH connections. Attackers may be able to intercept and decrypt sensitive information, leading to data breaches and security incidents. Ensuring strong encryption algorithms are used in SSH connections is essential to protect sensitive data and maintain the security of remote access.

Affected URL:
The affected URL or service is not applicable in this vulnerability as it pertains to the SSH server configuration rather than a specific web service.

Proof of Concept (PoC):
A Proof of Concept (PoC) is not provided in the given information, as this vulnerability is related to the configuration of the SSH server and the supported encryption algorithms. Exploiting such a vulnerability typically requires knowledge of the target server's SSH configuration and access to SSH services, which are not detailed in the plugin output.

Remediation:
To mitigate this vulnerability and enhance the security of SSH connections, follow these steps:

- **SSH Server Configuration:** Review and update the SSH server configuration to disallow the use of weak encryption algorithms such as Arcfour, Arcfour128, and Arcfour256.
- **Strong Encryption Algorithms:** Configure the SSH server to use strong encryption algorithms that are recommended for security, such as AES (Advanced Encryption Standard) or more robust alternatives.
- **Disable Weakened Algorithms:** Disable support for the Arcfour stream cipher and its variations in the SSH server's configuration.
- **Key Exchange Algorithms:** Ensure that the SSH server also employs secure key exchange algorithms to establish secure connections.
- **Regular Updates:** Keep the SSH server software and related libraries up-to-date to benefit from security improvements and patches.

- **Monitoring and Logging:** Implement monitoring and logging of SSH connection attempts and security events to detect and respond to any suspicious activities.
- **SSH Hardening:** Follow SSH hardening best practices, including disabling unnecessary services and implementing secure user authentication methods.
- **Security Policies:** Establish and enforce security policies that dictate the use of strong encryption algorithms and configurations for SSH connections.
- **Security Awareness:** Train administrators and users on secure SSH practices and the importance of using strong encryption.