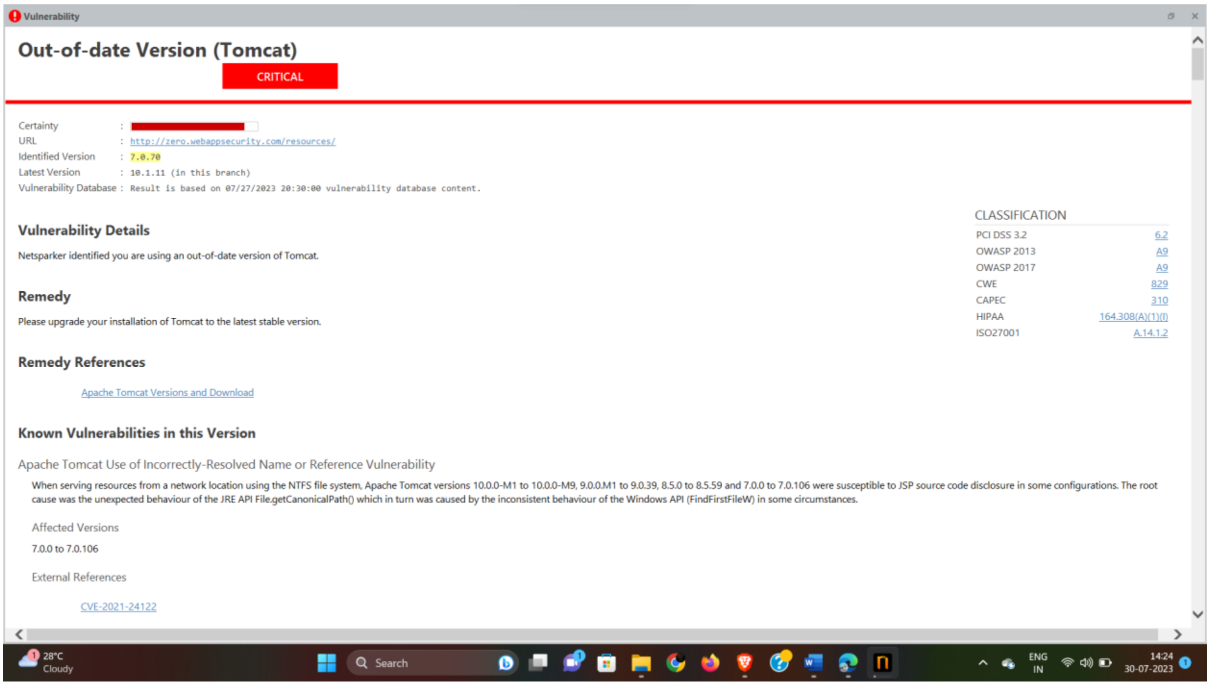
Task 2 :

3 critical vulnerabilities found in http://zero.webappsecurity.com/

Vulnerability 1 :



Out-of-date Version (Tomcat)

Certainty :
URL : http://zero.webappsecurity.com/resources/
Identified Version : 7.0.70
Latest Version : 10.1.11 (in this branch)
Vulnerability Database : Result is based on 07/27/2023 20:30:00 vulnerability database content.

CLASSIFICATION	
PCI DSS 3.2	6.2
OWASP 2013	A9
OWASP 2017	A9
CWE	829
CAPEC	310
HIPAA	164.308(A)(1)(I)
ISO27001	A.14.1.2

Vulnerability Details

Netsparker identified you are using an out-of-date version of Tomcat.

Remedy

Please upgrade your installation of Tomcat to the latest stable version.

Remedy References

Apache Tomcat Versions and Download

Known Vulnerabilities in this Version

Apache Tomcat Use of Incorrectly-Resolved Name or Reference Vulnerability

When serving resources from a network location using the NTFS file system, Apache Tomcat versions 10.0.0-M1 to 10.0.0-M9, 9.0.0.M1 to 9.0.39, 8.5.0 to 8.5.59 and 7.0.0 to 7.0.106 were susceptible to JSP source code disclosure in some configurations. The root cause was the unexpected behaviour of the JRE API File.getCanonicalPath() which in turn was caused by the inconsistent behaviour of the Windows API (FindFirstFileW) in some circumstances.

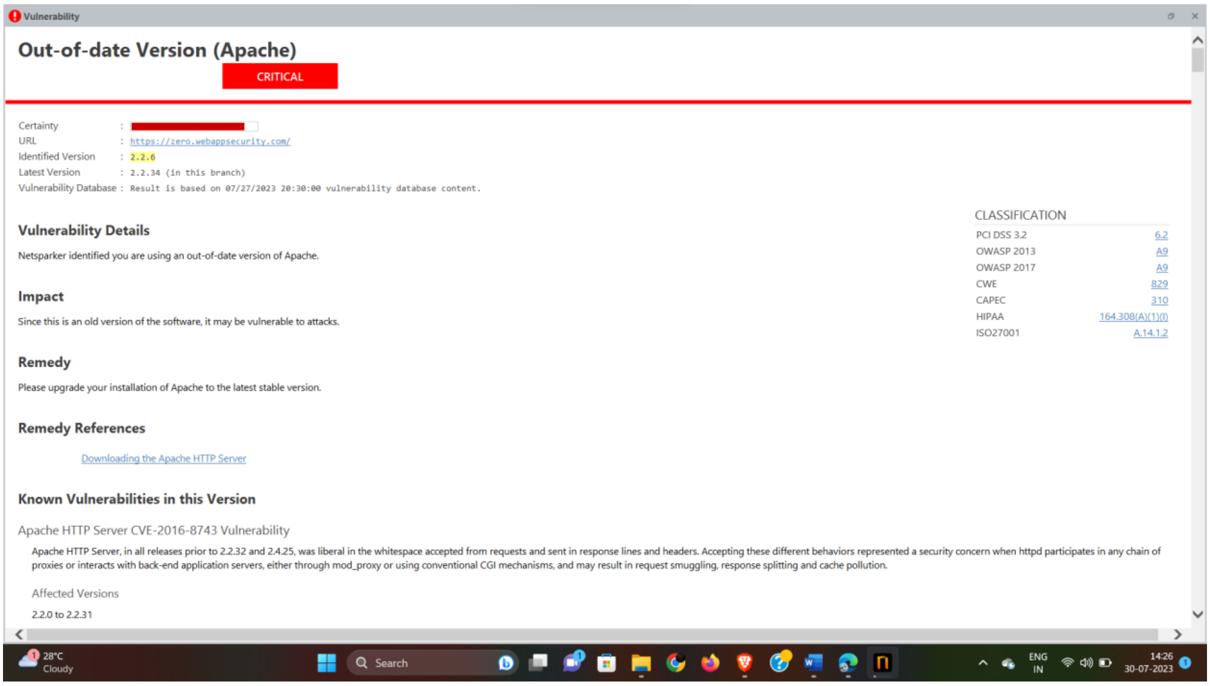
Affected Versions

7.0.0 to 7.0.106

External References

CVE-2021-24122

Vulnerability 2 :



Out-of-date Version (Apache)

Certainty :
URL : https://zero.webappsecurity.com/
Identified Version : 2.2.6
Latest Version : 2.2.34 (in this branch)
Vulnerability Database : Result is based on 07/27/2023 20:30:00 vulnerability database content.

CLASSIFICATION	
PCI DSS 3.2	6.2
OWASP 2013	A9
OWASP 2017	A9
CWE	829
CAPEC	310
HIPAA	164.308(A)(1)(I)
ISO27001	A.14.1.2

Vulnerability Details

Netsparker identified you are using an out-of-date version of Apache.

Impact

Since this is an old version of the software, it may be vulnerable to attacks.

Remedy

Please upgrade your installation of Apache to the latest stable version.

Remedy References

Downloading the Apache HTTP Server

Known Vulnerabilities in this Version

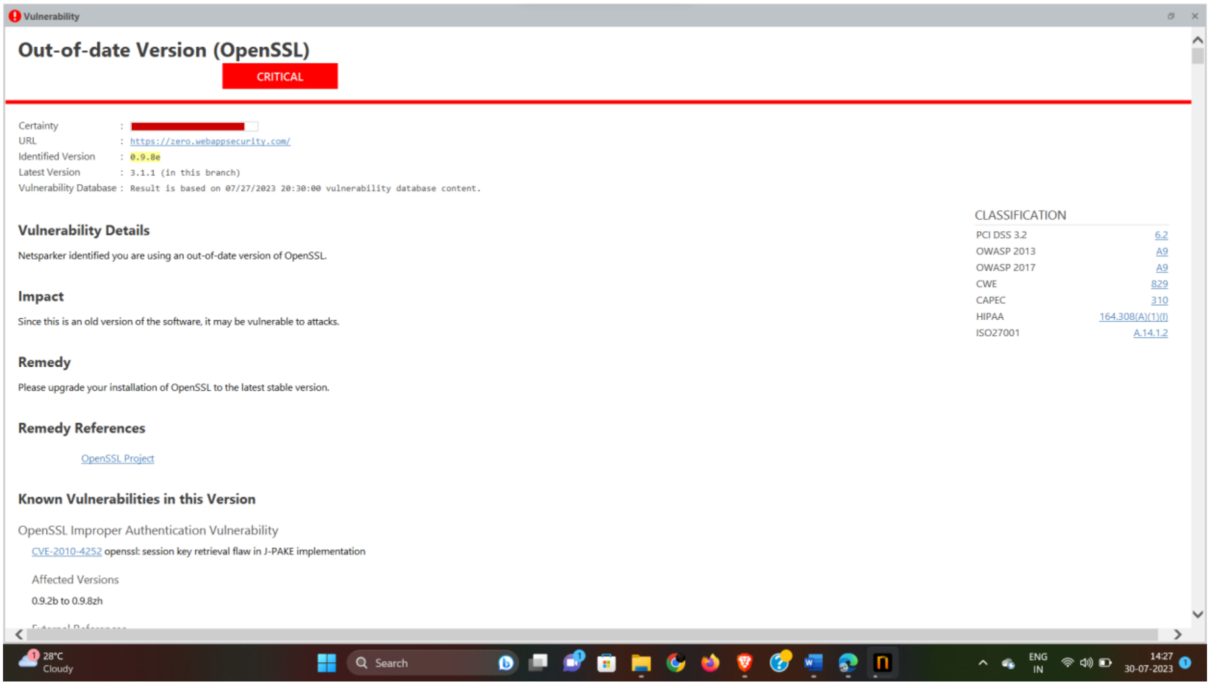
Apache HTTP Server CVE-2016-8743 Vulnerability

Apache HTTP Server, in all releases prior to 2.2.32 and 2.4.25, was liberal in the whitespace accepted from requests and sent in response lines and headers. Accepting these different behaviors represented a security concern when httpd participates in any chain of proxies or interacts with back-end application servers, either through mod_proxy or using conventional CGI mechanisms, and may result in request smuggling, response splitting and cache pollution.

Affected Versions

2.2.0 to 2.2.31

Vulnerability 3 :



Out-of-date Version (OpenSSL)

CRITICAL

Certainty	:
URL	: https://zero.webappsecurity.com/
Identified Version	: 0.9.8e
Latest Version	: 3.1.1 (in this branch)
Vulnerability Database	: Result is based on 07/27/2023 20:30:00 vulnerability database content.

Vulnerability Details

Netsparker identified you are using an out-of-date version of OpenSSL.

Impact

Since this is an old version of the software, it may be vulnerable to attacks.

Remedy

Please upgrade your installation of OpenSSL to the latest stable version.

Remedy References

OpenSSL Project

Known Vulnerabilities in this Version

OpenSSL Improper Authentication Vulnerability

CVE-2010-4252 openssl: session key retrieval flaw in J-PAKE implementation

Affected Versions

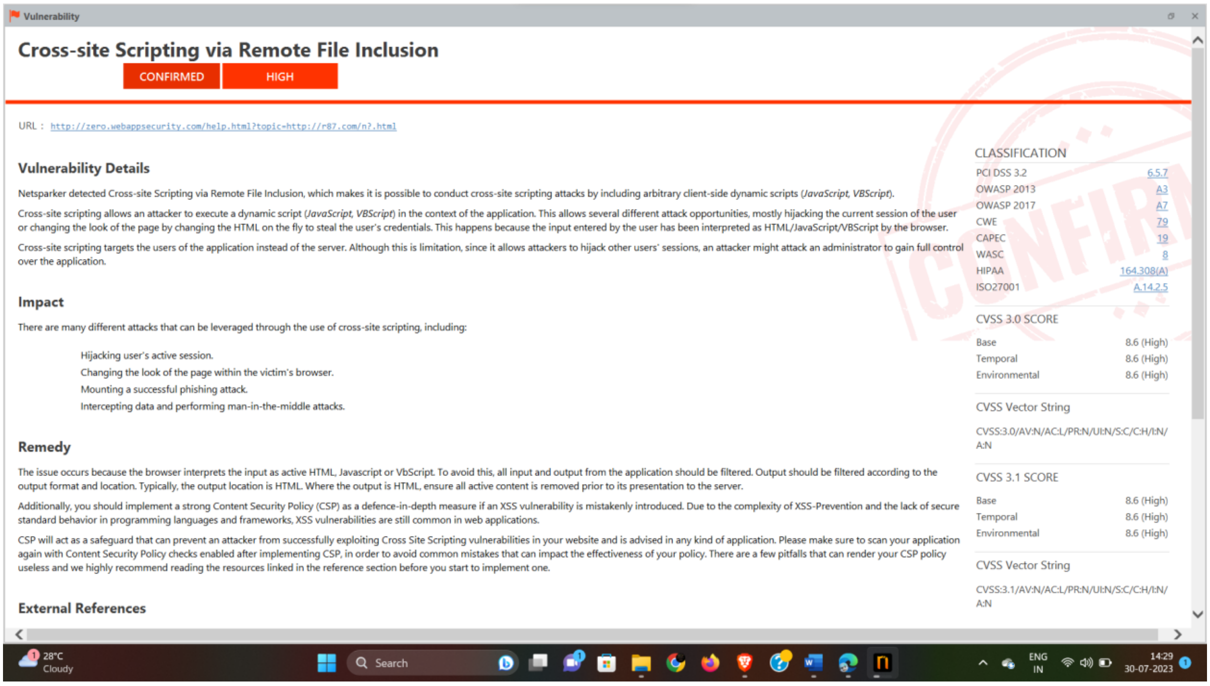
0.9.2b to 0.9.8zh

CLASSIFICATION

PCI DSS 3.2	6.2
OWASP 2013	A9
OWASP 2017	A9
CWE	829
CAPEC	310
HIPAA	164.308(A)(1)(I)
ISO27001	A.14.1.2

High risk Vulnerabilities :

Vulnerability 1 :



Cross-site Scripting via Remote File Inclusion

CONFIRMED HIGH

URL : http://zero.webappsecurity.com/help.html?topic=http://r87.com/n?.html

Vulnerability Details

Netsparker detected Cross-site Scripting via Remote File Inclusion, which makes it is possible to conduct cross-site scripting attacks by including arbitrary client-side dynamic scripts (*JavaScript, VBScript*).

Cross-site scripting allows an attacker to execute a dynamic script (*JavaScript, VBScript*) in the context of the application. This allows several different attack opportunities, mostly hijacking the current session of the user or changing the look of the page by changing the HTML on the fly to steal the user's credentials. This happens because the input entered by the user has been interpreted as HTML/JavaScript/VBScript by the browser.

Cross-site scripting targets the users of the application instead of the server. Although this is limitation, since it allows attackers to hijack other users' sessions, an attacker might attack an administrator to gain full control over the application.

Impact

There are many different attacks that can be leveraged through the use of cross-site scripting, including:

Hijacking user's active session.
Changing the look of the page within the victim's browser.
Mounting a successful phishing attack.
Intercepting data and performing man-in-the-middle attacks.

Remedy

The issue occurs because the browser interprets the input as active HTML, Javascript or VbScript. To avoid this, all input and output from the application should be filtered. Output should be filtered according to the output format and location. Typically, the output location is HTML. Where the output is HTML, ensure all active content is removed prior to its presentation to the server.

Additionally, you should implement a strong Content Security Policy (CSP) as a defence-in-depth measure if an XSS vulnerability is mistakenly introduced. Due to the complexity of XSS-Prevention and the lack of secure standard behavior in programming languages and frameworks, XSS vulnerabilities are still common in web applications.

CSP will act as a safeguard that can prevent an attacker from successfully exploiting Cross Site Scripting vulnerabilities in your website and is advised in any kind of application. Please make sure to scan your application again with Content Security Policy checks enabled after implementing CSP, in order to avoid common mistakes that can impact the effectiveness of your policy. There are a few pitfalls that can render your CSP policy useless and we highly recommend reading the resources linked in the reference section before you start to implement one.

External References

CLASSIFICATION

PCI DSS 3.2	6.5.7
OWASP 2013	A3
OWASP 2017	A7
CWE	79
CAPEC	19
WASC	8
HIPAA	164.308(A)
ISO27001	A.14.2.5

CVSS 3.0 SCORE

Base	8.6 (High)
Temporal	8.6 (High)
Environmental	8.6 (High)

CVSS Vector String

CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:N/A:N

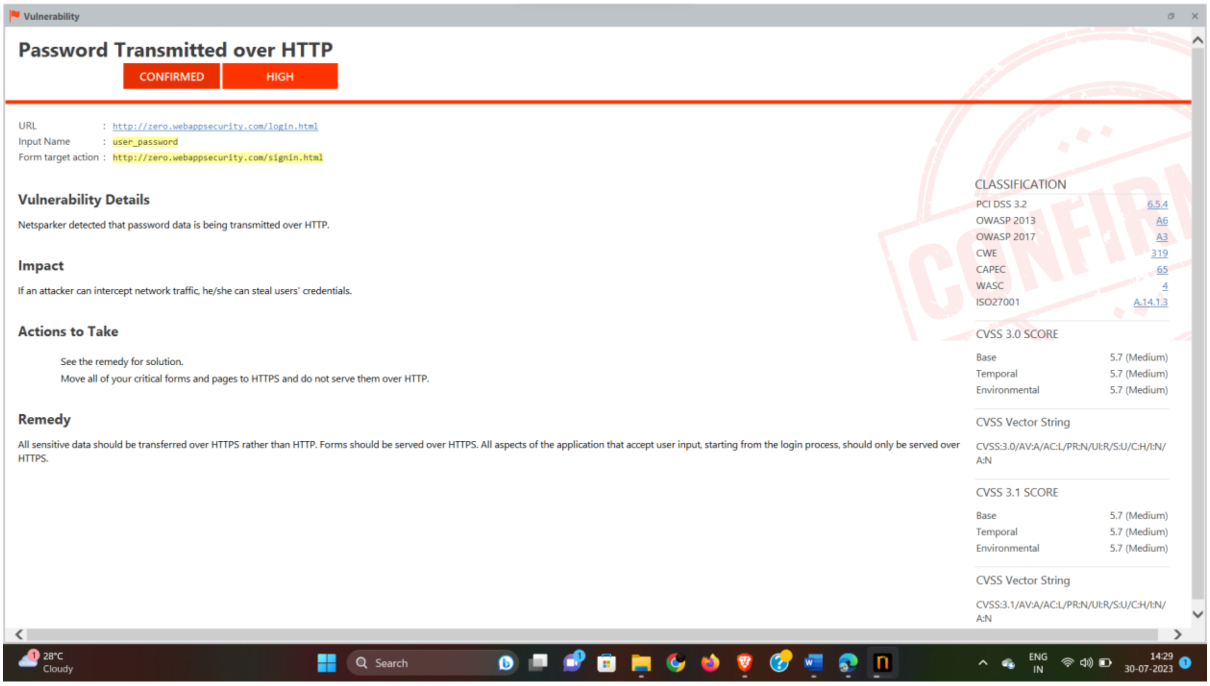
CVSS 3.1 SCORE

Base	8.6 (High)
Temporal	8.6 (High)
Environmental	8.6 (High)

CVSS Vector String

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:N/A:N

Vulnerability 2 :



Password Transmitted over HTTP

URL : http://zero.webappsecurity.com/login.html
Input Name : user_password
Form target action : http://zero.webappsecurity.com/signin.html

Vulnerability Details

Netsparker detected that password data is being transmitted over HTTP.

Impact

If an attacker can intercept network traffic, he/she can steal users' credentials.

Actions to Take

See the remedy for solution.
Move all of your critical forms and pages to HTTPS and do not serve them over HTTP.

Remedy

All sensitive data should be transferred over HTTPS rather than HTTP. Forms should be served over HTTPS. All aspects of the application that accept user input, starting from the login process, should only be served over HTTPS.

CLASSIFICATION	
PCI DSS 3.2	6.5.4
OWASP 2013	A6
OWASP 2017	A3
CWE	319
CAPEC	65
WASC	4
ISO27001	A.14.1.3

CVSS 3.0 SCORE	
Base	5.7 (Medium)
Temporal	5.7 (Medium)
Environmental	5.7 (Medium)

CVSS Vector String

CVSS:3.0/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N

CVSS 3.1 SCORE	
Base	5.7 (Medium)
Temporal	5.7 (Medium)
Environmental	5.7 (Medium)

CVSS Vector String

CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N

My Report :

Report Title : Password Transmitted over HTTP

IDOR on http://zero.webappsecurity.com/ leads to takeover of user credentials

Report Summary :

Here the website uses HTTP which is not quit secure , hence which may lead to different vulnerabilities. So these one the High Vulnerability in the login page of the website where the user credentials can be stolen.

URL : http://zero.webappsecurity.com/login.html

Input Name : user_password

Vulnerability Details :

Password transmission over HTTP is a serious security hole that exposes users' sensitive data, including login credentials. Since the data supplied through the HTTP (Hypertext Transfer Protocol) protocol is not encrypted, anyone with the necessary access and means can intercept and read the data being transmitted, including passwords.Passwords are transferred in plaintext over HTTP when they are transmitted by a bank's login page or any other sensitive service. The communication between the user's web browser and the bank's server can then be overheard by attackers. The unencrypted password data can be intercepted by the attackers using a variety of strategies, including man-in-the-middle attacks, packet sniffing, or network monitoring.

Consequences / Impact :

Password Compromise: Attackers can readily get user passwords, enabling them access to user accounts without authorization.

Identity theft: If users use the same passwords across other platforms, hackers may gain access to other accounts, such as email, social networking, or online shopping, using the stolen information.

Financial Loss: If an attack were to occur on a banking website, money might be taken from user accounts or transactions could be made without authorization.

Privacy Breach: Attackers have access to other sensitive data sent over HTTP, jeopardizing user privacy.

Avoidance and Remedy :

The bank or any service provider should put the following security measures into place to address the "Password Transmitted over HTTP" vulnerability:

Utilize HTTPS Make sure that all web pages on the bank's website use HTTPS (Hypertext Transfer Protocol Secure), especially those that deal with sensitive data like login passwords. It becomes significantly more difficult for attackers to intercept and decode the data transferred through HTTPS between the user's browser and the server.

HTTP Strict Transport Security, often known as HSTS: Force all contact with the server to happen over HTTPS by using HTTP Strict Transport Security. By doing this, visitors are prevented from unintentionally using an unsecured HTTP connection to view the website.

Password encryption: Before saving user passwords in the database, make sure that they are securely hashed and salted on the server. In the event that the database is hacked, this offers an extra degree of security.

Encourage or mandate users to adopt multi-factor authentication (MFA), which provides an additional degree of protection beyond simply a password.

Security Awareness Training: Inform both staff members and consumers of the value of using secure passwords, to refrain from reusing them, and to be wary of phishing scams.

Regular Security Audits: Conduct regular security audits to find and quickly fix any possible flaws.