Sri Lanka Institute of Information Technology

Bug Bounty Report 03

winscorecard2.aswatson.com

IE2062 – Web Security

Submitted by:

IT22227836 – ADHIKARI A.M.V.B.M

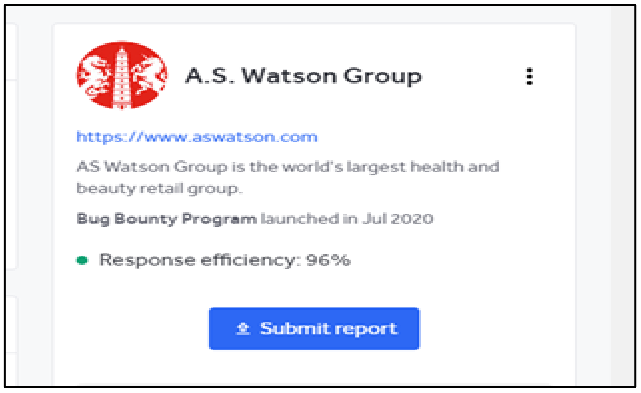
Date of submission

2024.10.31

Table of Contents

Report 03 – winscorecard2.aswatson.com

Main domain	https://www.aswatson.com/
Sub domain	winscorecard2.aswatson.com
IP address	210.0.245.141
platform	HackerOne


A.S. Watson Group

https://www.aswatson.com

AS Watson Group is the world's largest health and beauty retail group.

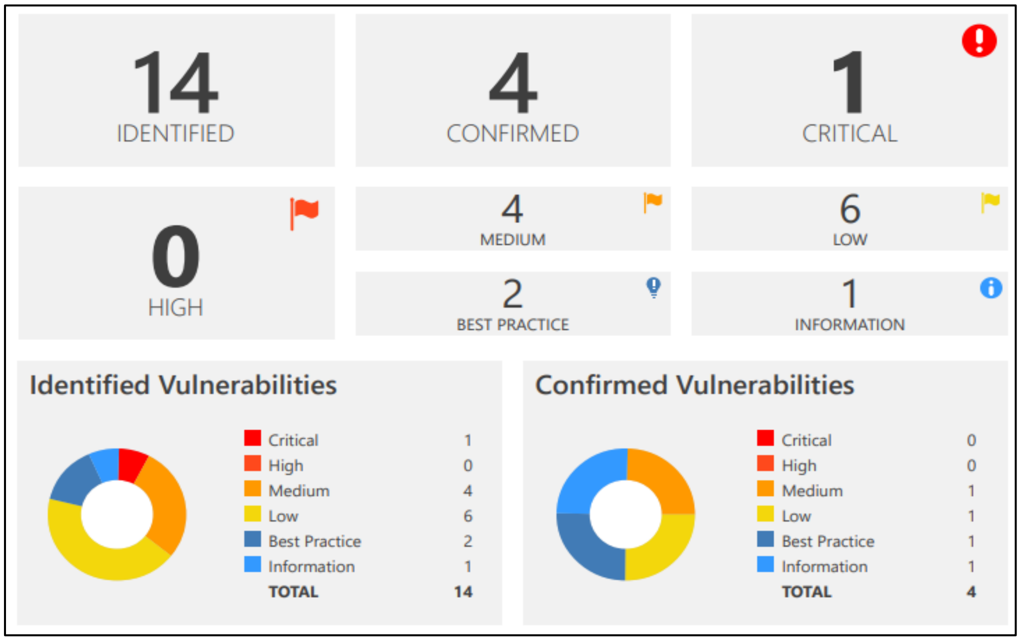
Bug Bounty Program launched in Jul 2020

● Response efficiency: 96%

The A.S. Watson Group is the world's largest health and beauty retail group, with over 15,700 stores in 25 markets worldwide serving over 28 million customers per week, and over 3 billion customers and members.

A.S. Watson Group looks forward to working with the security community to discover vulnerabilities to keep our businesses and customers safe. As we operate in many different countries, we will be rolling out our bug bounty program in phases. Our focus within this rollout, is our retail websites (and soon, mobile apps on both Android and IOS)

Vulnerability detected



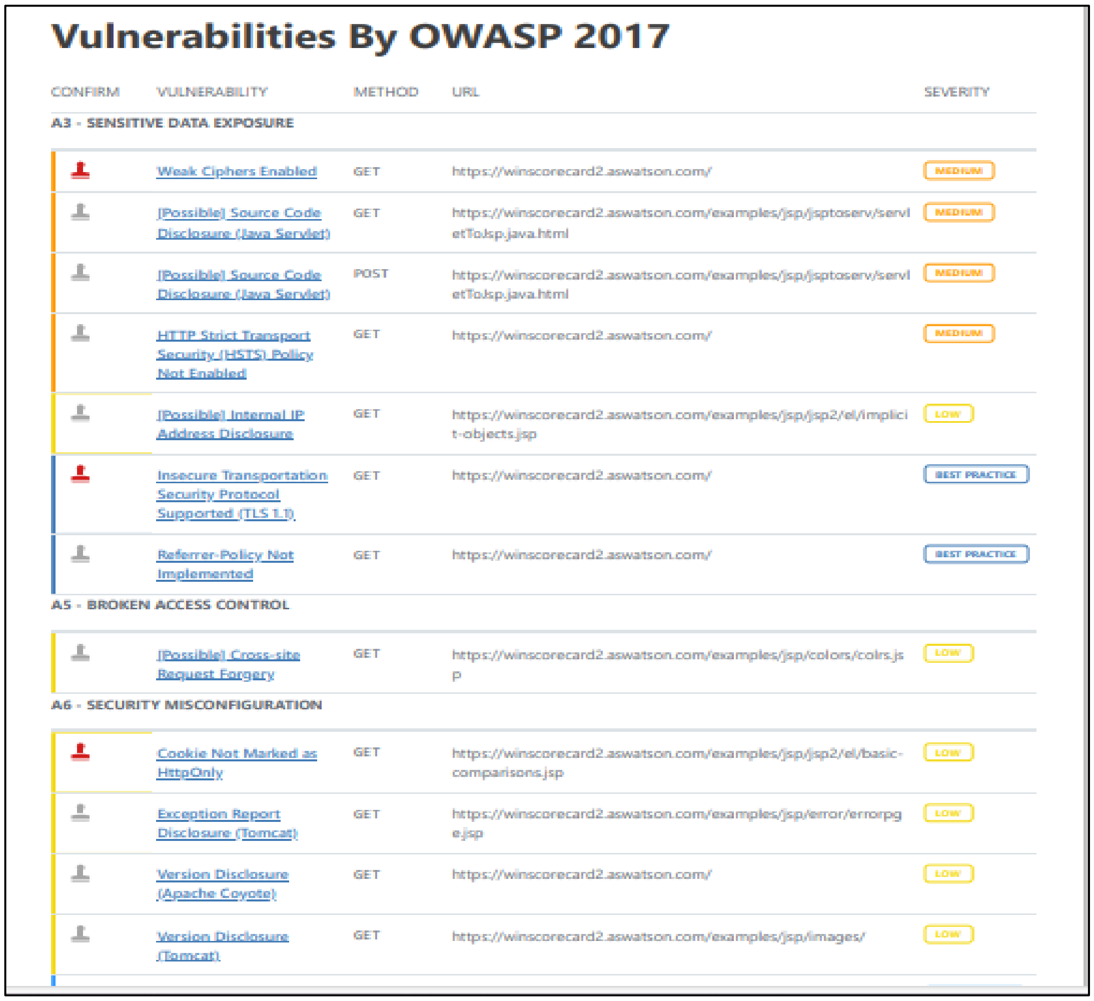
14 IDENTIFIED	4 CONFIRMED	1 CRITICAL
0 HIGH	4 MEDIUM	6 LOW
	2 BEST PRACTICE	1 INFORMATION

Identified Vulnerabilities

🟥	Critical	1
🟧	High	0
🟧	Medium	4
🟨	Low	6
🟦	Best Practice	2
🟦	Information	1
	TOTAL	**14**

Confirmed Vulnerabilities

🟥	Critical	0
🟧	High	0
🟧	Medium	1
🟨	Low	1
🟦	Best Practice	1
🟦	Information	1
	TOTAL	**4**

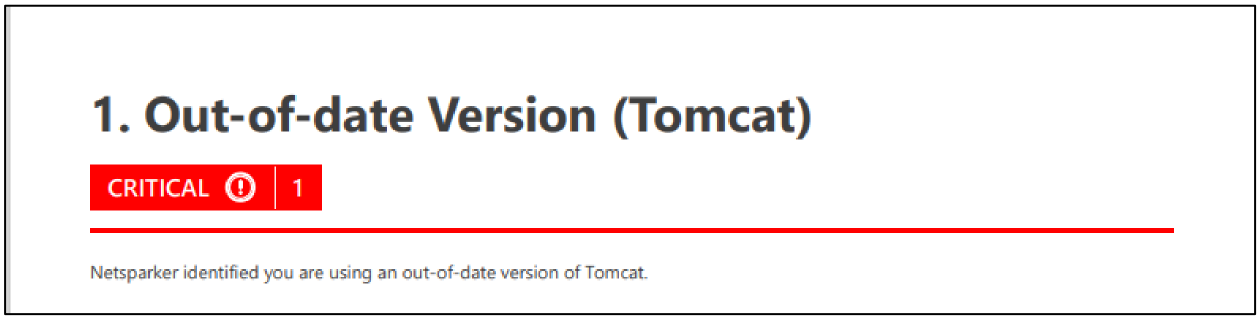


Vulnerabilities By OWASP 2017

CONFIRM	VULNERABILITY	METHOD	URL	SEVERITY
A3 - SENSITIVE DATA EXPOSURE				
✓	Weak Ciphers Enabled	GET	https://winscorecard2.aswatson.com/	MEDIUM
	[Possible] Source Code Disclosure (Java Servlet)	GET	https://winscorecard2.aswatson.com/examples/jsp/jsptoserv/servletToJsp.java.html	MEDIUM
	[Possible] Source Code Disclosure (Java Servlet)	POST	https://winscorecard2.aswatson.com/examples/jsp/jsptoserv/servletToJsp.java.html	MEDIUM
	HTTP Strict Transport Security (HSTS) Policy Not Enabled	GET	https://winscorecard2.aswatson.com/	MEDIUM
	[Possible] Internal IP Address Disclosure	GET	https://winscorecard2.aswatson.com/examples/jsp/jsp2/el/implicit-objects.jsp	LOW
✓	Insecure Transportation Security Protocol Supported (TLS 1.1)	GET	https://winscorecard2.aswatson.com/	BEST PRACTICE
	Referrer-Policy Not Implemented	GET	https://winscorecard2.aswatson.com/	BEST PRACTICE
A5 - BROKEN ACCESS CONTROL				
	[Possible] Cross-site Request Forgery	GET	https://winscorecard2.aswatson.com/examples/jsp/colors/colrs.jsp	LOW
A6 - SECURITY MISCONFIGURATION				
✓	Cookie Not Marked as HttpOnly	GET	https://winscorecard2.aswatson.com/examples/jsp/jsp2/el/basic-comparisons.jsp	LOW
	Exception Report Disclosure (Tomcat)	GET	https://winscorecard2.aswatson.com/examples/jsp/error/errorpge.jsp	LOW
	Version Disclosure (Apache Coyote)	GET	https://winscorecard2.aswatson.com/	LOW
	Version Disclosure (Tomcat)	GET	https://winscorecard2.aswatson.com/examples/jsp/images/	LOW

Vulnerability

1. Title - Out-of-date Version (Tomcat)



1. Out-of-date Version (Tomcat)

CRITICAL 1

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

- Risk – Critical

Description

Out-of-date Version-Tomcat: This vulnerability involves a web server running an older version of Apache Tomcat, which has become unsupported or was not recently patched with the latest security updates. The general problem with older versions lies in the various well-known bugs; this means that knowledge about them is publicly available, and it is far easier for an attacker to use them. Such bugs can pose serious security hazards, such as remote code execution, denial of service issues, or even unauthorized data disclosure.

Keeping Apache Tomcat updated means keeping it secure. The periodic updates to the latest stable version ensure that known vulnerabilities are patched, thereby reducing the chances of successful exploitation. Besides, good awareness about security advisories and quick application of security patches once released will save the web application and its users from potential attacks.

Affected components

Here, the Out-of-date Version can be realized in the following components of the Tomcat vulnerability:

- **Web Applications Running on Tomcat:**
  Applications running on an outdated version of the Tomcat Server expose themselves to exploits due to known vulnerabilities. These may lead to information disclosure, remote code execution, or unauthorized access.

- **Tomcat Server Configuration:**
  Configurations of the server and management interfaces such as /manager and /admin may also be attacked, and this may provide an attacker with the opportunity to change server settings or take administrative control.

- **Underlying Operating System:**
  In many cases, an attacker uses the exploitation of a Tomcat server as a steppingstone toward an underlying operating system for systemic compromise.

- **SSL/TLS Implementation:**
  Older versions will have insecure implementations of SSL/TLS, exposing secure communications to various vulnerabilities such as weak encryption or Man-in-the-Middle attacks.

- **Connected Databases and Services:**
  Tomcat vulnerabilities, if exploited, may potentially allow attackers to reach databases and other connected backend services to the server for leakage or manipulation of data.

Keeping these updated and secured is of the essence in reducing risks associated with running an outdated version of Tomcat.

Impact Assessment

Running an outdated version of Apache Tomcat involves serious security consequences and operational impact, including but not limited to the following.

- **Security Breaches:**
  - RCE: An attacker might use the vulnerabilities that older versions possess to successfully execute arbitrary code in order to have full control over the server.
  - Data Exposure: An attacker may view or change sensitive data, such as customer information or application secrets.
  - Privilege Escalation: Exploiting such known flaws could enable attackers to escalate privileges and thereby gain access to key system resources.
- **Service Disruption:**
  Denial of Service: In some older versions, it might be vulnerable to such attacks that overload the server; this makes the web applications unavailable to legitimate users and results in downtime and loss of service.
- **Reputation Damage:**
  A data breach or successful attack owing to running an older version of Tomcat may result in the loss of customer, partner, and stakeholder trust, thereby compromising the brand reputation of an organization.
- **Compliance Violation:**
  Using very old software could result in failure to meet security standards and regulations, such as GDPR and PCI-DSS, which in turn may bring fines or other judgments.
- **Financial Loss:**
  A data breach caused by unpatched vulnerabilities may lead to massive financial loss, composited of costs related to incident response, litigation, and loss of business.

Considering these risks, one of the main precautions is to upgrade Apache Tomcat to its most stable version, so its security and integrity, along with those of the web applications dependent on it, are preserved.

Steps to reproduce

1. **Identify the Target Server:**
    - Confirm that the target system is running Apache Tomcat.

2. **Check the Tomcat Version:**
    - Access the server's HTTP headers or admin interface to find the version of Tomcat being used. The version is often displayed in the HTTP response headers or visible in the admin or manager interface of Tomcat

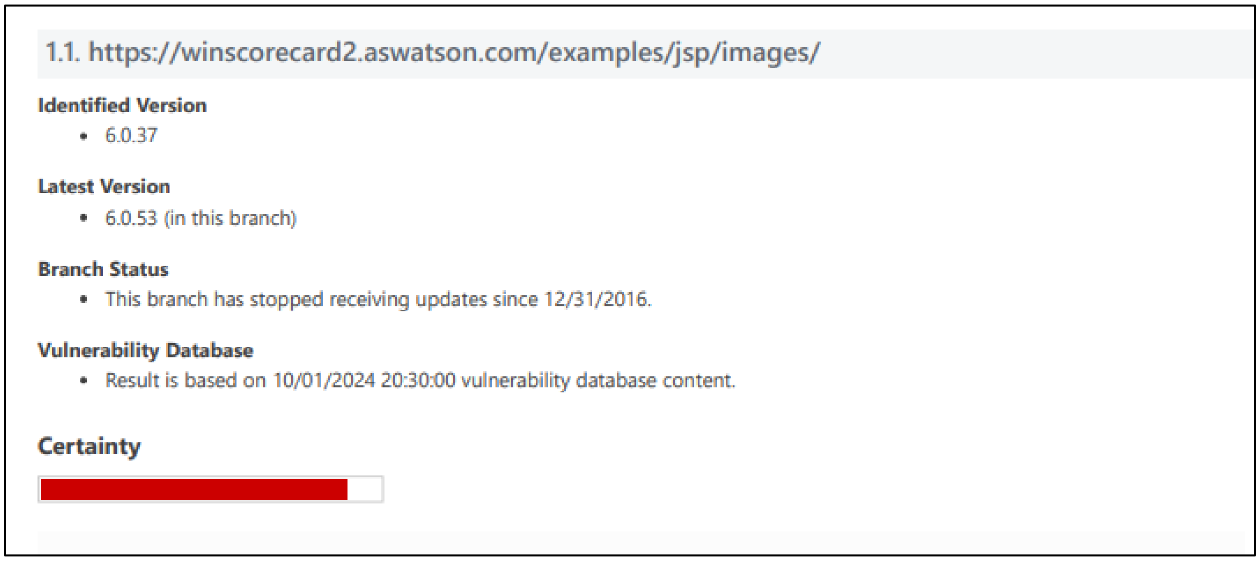
3. **Compare with Latest Version:**
    - Cross-check the identified Tomcat version with the latest stable version available on the official Apache Tomcat website. Verify if the current version is outdated.

4. **Review Known Vulnerabilities:**
    - Search for vulnerabilities associated with the outdated Tomcat version on security databases like the National Vulnerability Database (NVD) or CVE Details to assess its risks.

5. **Document the Findings:**
    - If the version is outdated and vulnerable, note down the version number and the known vulnerabilities for inclusion in your report.

Proof of concept (if applicable)

Vulnerability scanning using Netsparker



1.1. https://winscorecard2.aswatson.com/examples/jsp/images/

Identified Version

- 6.0.37

Latest Version

- 6.0.53 (in this branch)

Branch Status

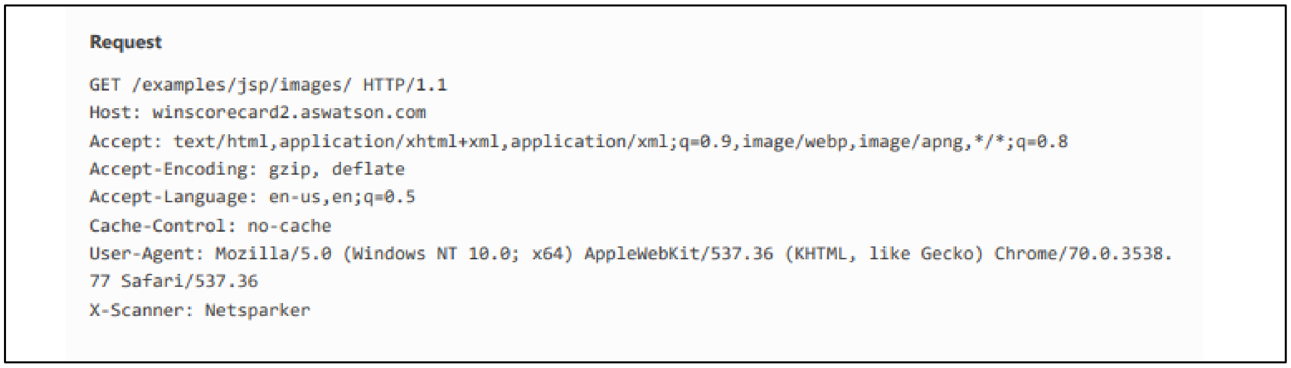
- This branch has stopped receiving updates since 12/31/2016.

Vulnerability Database

- Result is based on 10/01/2024 20:30:00 vulnerability database content.

Certainty

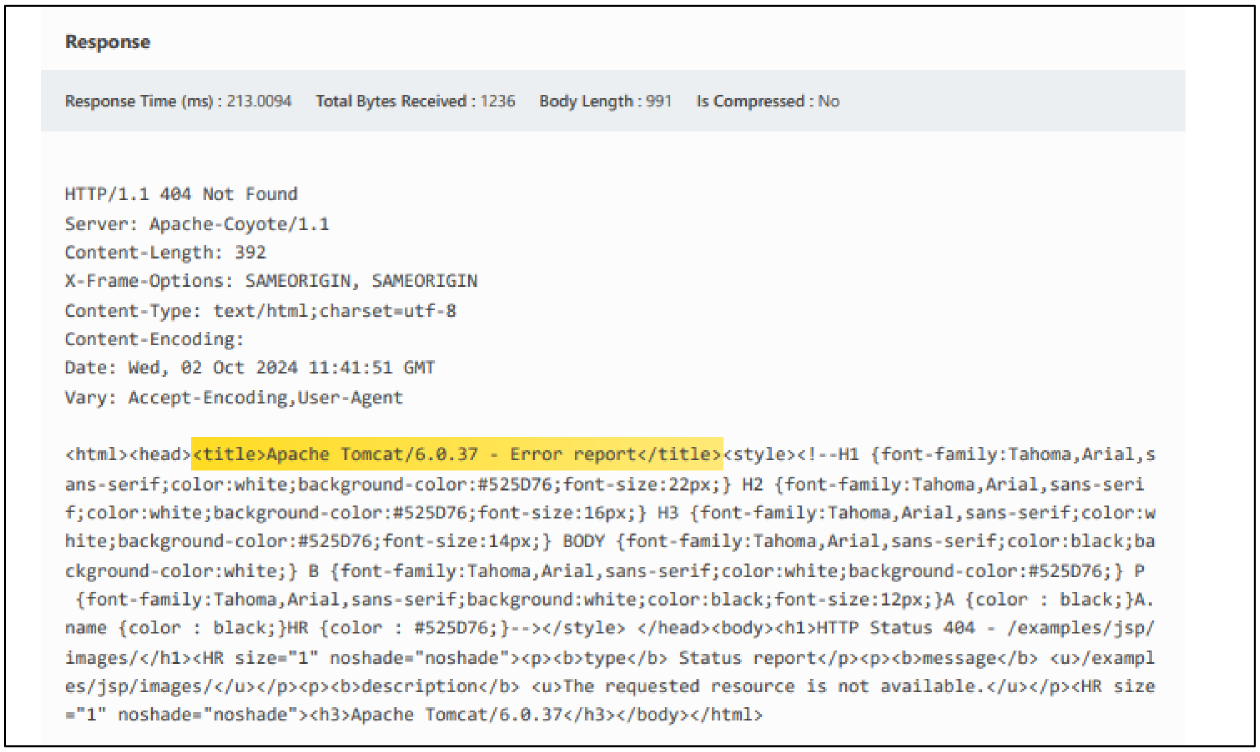
- **Request**



Request

```
GET /examples/jsp/images/ HTTP/1.1
Host: winscorecard2.aswatson.com
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-us,en;q=0.5
Cache-Control: no-cache
User-Agent: Mozilla/5.0 (Windows NT 10.0; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.
77 Safari/537.36
X-Scanner: Netsparker
```

- **Response**



Response

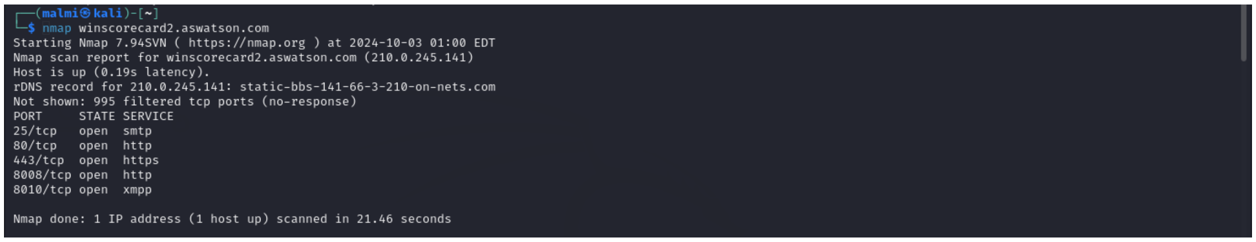
Response Time (ms) : 213.0094 Total Bytes Received : 1236 Body Length : 991 Is Compressed : No

```
HTTP/1.1 404 Not Found
Server: Apache-Coyote/1.1
Content-Length: 392
X-Frame-Options: SAMEORIGIN, SAMEORIGIN
Content-Type: text/html;charset=utf-8
Content-Encoding:
Date: Wed, 02 Oct 2024 11:41:51 GMT
Vary: Accept-Encoding,User-Agent

<html><head><title>Apache Tomcat/6.0.37 - Error report</title><style><!--H1 {font-family:Tahoma,Arial,s
ans-serif;color:white;background-color:#525D76;font-size:22px;} H2 {font-family:Tahoma,Arial,sans-seri
f;color:white;background-color:#525D76;font-size:16px;} H3 {font-family:Tahoma,Arial,sans-serif;color:w
hite;background-color:#525D76;font-size:14px;} BODY {font-family:Tahoma,Arial,sans-serif;color:black;ba
ckground-color:white;} B {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;} P
 {font-family:Tahoma,Arial,sans-serif;background:white;color:black;font-size:12px;}A {color : black;}A.
name {color : black;}HR {color : #525D76;}--></style> </head><body><h1>HTTP Status 404 - /examples/jsp/
images/</h1><HR size="1" noshade="noshade"><p><b>type</b> Status report</p><p><b>message</b> <u>/exampl
es/jsp/images/</u></p><p><b>description</b> <u>The requested resource is not available.</u></p><HR size
="1" noshade="noshade"><h3>Apache Tomcat/6.0.37</h3></body></html>
```

<u>Vulnerability finding using namp</u>

Used command

- nmap winscorecard2.aswatson.com



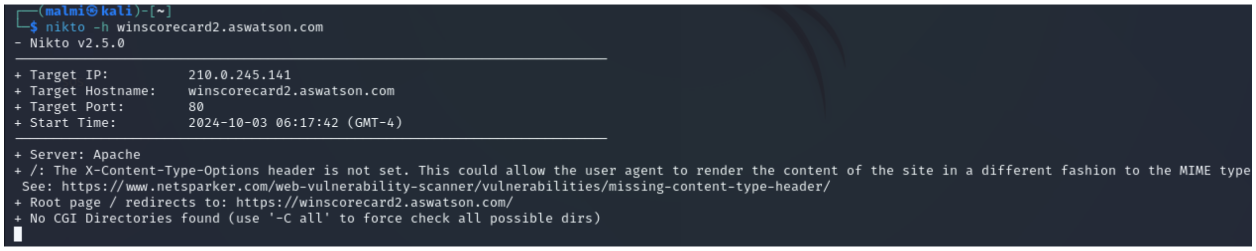
```
┌──(malmi㉿kali)-[~]
└─$ nmap winscorecard2.aswatson.com
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-03 01:00 EDT
Nmap scan report for winscorecard2.aswatson.com (210.0.245.141)
Host is up (0.19s latency).
rDNS record for 210.0.245.141: static-bbs-141-66-3-210-on-nets.com
Not shown: 995 filtered tcp ports (no-response)
PORT     STATE SERVICE
25/tcp   open  smtp
80/tcp   open  http
443/tcp  open  https
8008/tcp open  http
8010/tcp open  xmpp

Nmap done: 1 IP address (1 host up) scanned in 21.46 seconds
```

Founded open port

- 25/tcp open smtp
- 80/tcp open http
- 443/tcp open https
- 8008/tcp open http
- 8010/tcp open xmpp

<u>Vulnerability finding using Nikto</u>

- **Used command**
  - nikto -h winscorecard2.aswatson.com

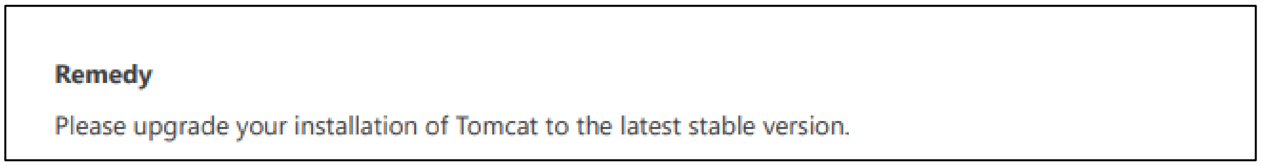


```
┌──(malmi㉿kali)-[~]
└─$ nikto -h winscorecard2.aswatson.com
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          210.0.245.141
+ Target Hostname:    winscorecard2.aswatson.com
+ Target Port:        80
+ Start Time:         2024-10-03 06:17:42 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
 See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ Root page / redirects to: https://winscorecard2.aswatson.com/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
```

Result:

The Nikto scan indicates that the server is running Apache, which means you will need to check if Apache or any applications running on it, such as Tomcat, are outdated.

Proposed mitigation or fix


Remedy

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

Go through the following proposed mitigations or fixes to address the Out-of-date Version vulnerability in Tomcat:

- Upgrade Tomcat
  - Upgrade to Latest Stable Version: The Tomcat server should be updated to the latest stable version. It may be able to resolve known vulnerabilities, hence improving security features regularly. Check updates and patch any identified vulnerabilities

- Apply Security Patches
  - Periodically look out for the security patches published by the ASF for Tomcat, along with upgrading. Apply those patches immediately to reduce the risks for vulnerabilities.

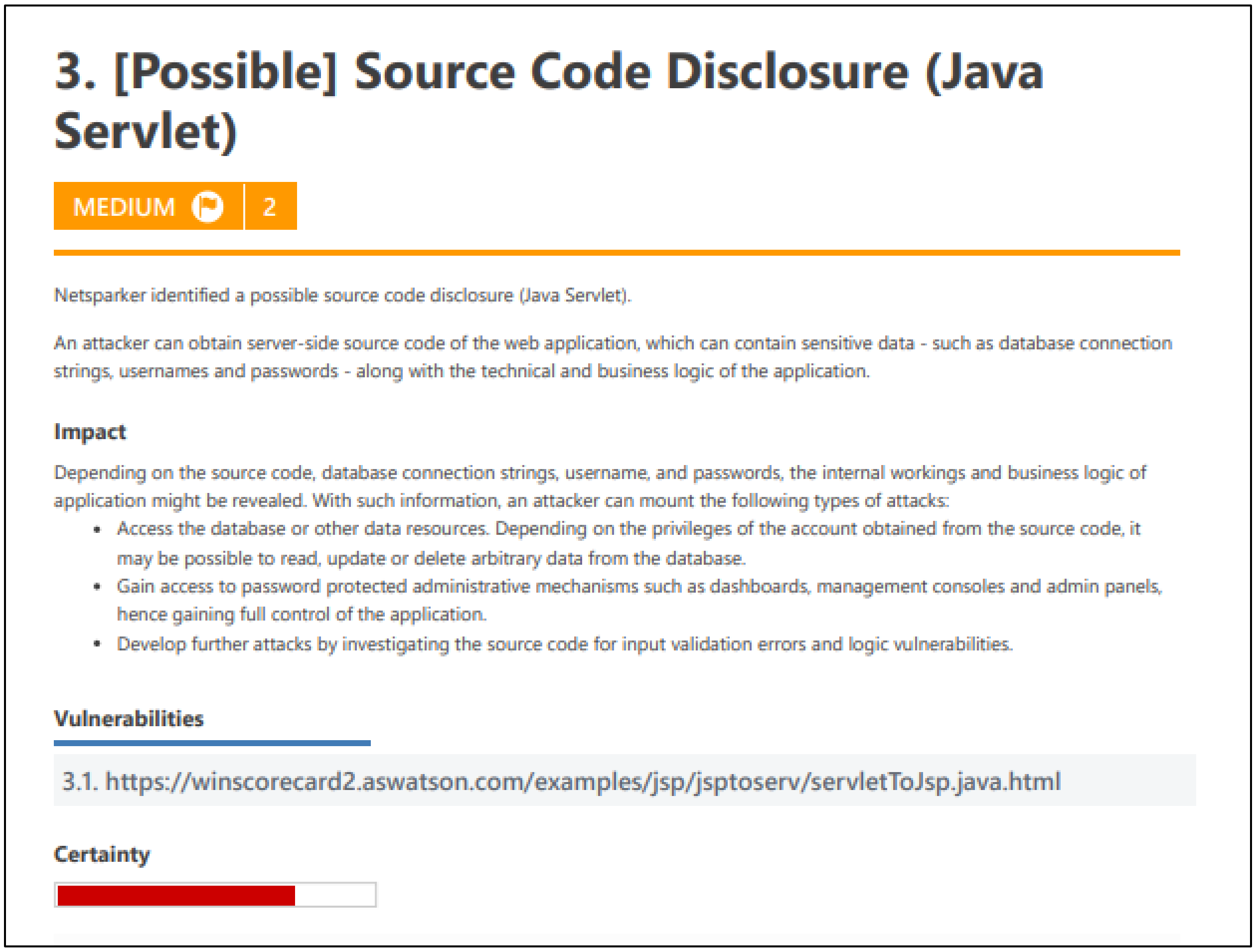
- Configure Security Headers
  - Apply Security Headers: One should implement the mechanism of security headers like X-Content-Type-Options, Content-Security-Policy, and X-Frame-Options. These headers will help in mitigating different types of attacks, such as sniffing of content type and clickjacking

- Proper Configuration of Access Controls
  - Limit Access to Sensitive Areas: Access to the Tomcat manager and other sensitive interfaces must be narrowed down. Strong authentication is enforced with only trusted IP addresses allowed access.

- Run Periodic Security Scans
  - Conduct Periodic Security Assessments: Web applications and server configurations must be scanned and audited frequently in order to find vulnerabilities and misconfigurations. Engage in thorough assessments using Nikto, Nessus, or OpenVAS.

- Utilize Application Firewalls
  - Deploy Web Application Firewalls: WAFs monitor and filter the incoming flow of traffic to your applications, providing a different layer of protection against common attacks.

2. Tittle - [Possible] Source Code Disclosure (Java Servlet)



3. [Possible] Source Code Disclosure (Java Servlet)

MEDIUM | 2

Netsparker identified a possible source code disclosure (Java Servlet).

An attacker can obtain server-side source code of the web application, which can contain sensitive data - such as database connection strings, usernames and passwords - along with the technical and business logic of the application.

Impact

Depending on the source code, database connection strings, username, and passwords, the internal workings and business logic of application might be revealed. With such information, an attacker can mount the following types of attacks:

- Access the database or other data resources. Depending on the privileges of the account obtained from the source code, it may be possible to read, update or delete arbitrary data from the database.
- Gain access to password protected administrative mechanisms such as dashboards, management consoles and admin panels, hence gaining full control of the application.
- Develop further attacks by investigating the source code for input validation errors and logic vulnerabilities.

Vulnerabilities

3.1. https://winscorecard2.aswatson.com/examples/jsp/jsptoserv/servletToJsp.java.html

Certainty

Description

It involves the disclosure of source code in web applications when an attacker gains access to critical files, showing the backlines of the code in the application. This may be due to misconfigurations around web servers, bad permission settings on files, or inferior coding. This is of especial importance to the applications of Java Servlet, which are Java applications running on the server-side that handle HTTP requests. Weaker access controls might end with disclosures of .java or .class files. Exposition of such files may reveal sensitive logic, processes of user input, or security mechanisms, like authentication and interaction with the database, which might all turn out valuable insights for subsequent attacks.

For example, an attacker may ask for Java source code files either by directly using the servlet paths or by trying directory traversal. Sometimes the web server is not well configured, and it serves the raw source code-the attacker can then read the application logic and discover other vulnerabilities.

Affected components

- **Servlet source code files**: several .java files, which define the behavior of Java Servlets. These may contain the logic for the application.
- **Compiled Java bytecode:** Sometimes, one is even able to decompile the .class files containing the compiled Java code into original source code.
- **Configuration Files:** These are files such as web.xml and context.xml that are used to manage servlet mappings, resource definitions, and application configurations.
- **Web Configuration:** Poor configurations in web servers like Apache Tomcat or Jetty are responsible for displaying protected directories.

For instance, directories like /WEB-INF and /META-INF are supposed not to be accessible with URLs directly. However, when the web server is not well-configured, an attacker can get access to those kinds of directories to view source codes or other sensitive information.

Impact Assessment

- **Entering source code:** This provides an attacker having full, inside knowledge regarding the functioning of the application, including proprietary business logic, credentials, API keys, and more.
- **Larger attack surface:** With source code, an attacker is better able to understand how the application interacts with its database, authentication mechanisms, and security measures. That might enable second-order attacks like SQL injection, command injection, or privilege escalation.
- **Intellectual Property Theft:** Proprietary code disclosure is oftentimes intellectual property loss when the code happens to be unique or part of core functionality of an application.
- **Reputational Damage:** If the customers or clients perceive the event of source code disclosure, the potential consequences of data breaches or other types of security compromises may make them lose trust in the organization.

Basically, active exploitation of this vulnerability would mean that the attacker would be able to gain a deep understanding of the system architecture, security measures implemented, and other potential vulnerabilities to use.

Steps to reproduce

- **Identify the Target Application:**
  Determine a Java-based web application that might have vulnerabilities exposing source code, such as files or directories related to Java classes or configuration files.

- **Access Potentially Sensitive Directories:**
  Attempt to access common sensitive directories such as /WEB-INF/ or /META-INF/ using the browser or manual URL input (e.g., http://<target>/WEB-INF/web.xml).

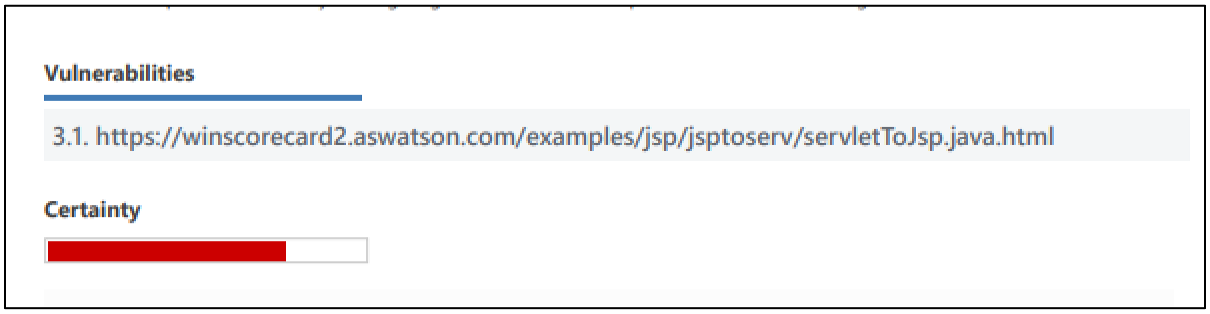
- **Direct File Access:**
  Try accessing specific Java source files or configuration files directly through URLs, for example, http://<target>/WEB-INF/classes/Servlet.java.

- **Check for Directory Browsing:**
  Explore whether directory browsing is enabled by attempting to visit sensitive directories (e.g., /WEB-INF/). This may reveal a list of files, including Java source code or other confidential data.

By following these steps, you could reproduce the vulnerability and observe if the server exposes sensitive information or files that shouldn't be publicly accessible.

Proof of concept (if applicable)

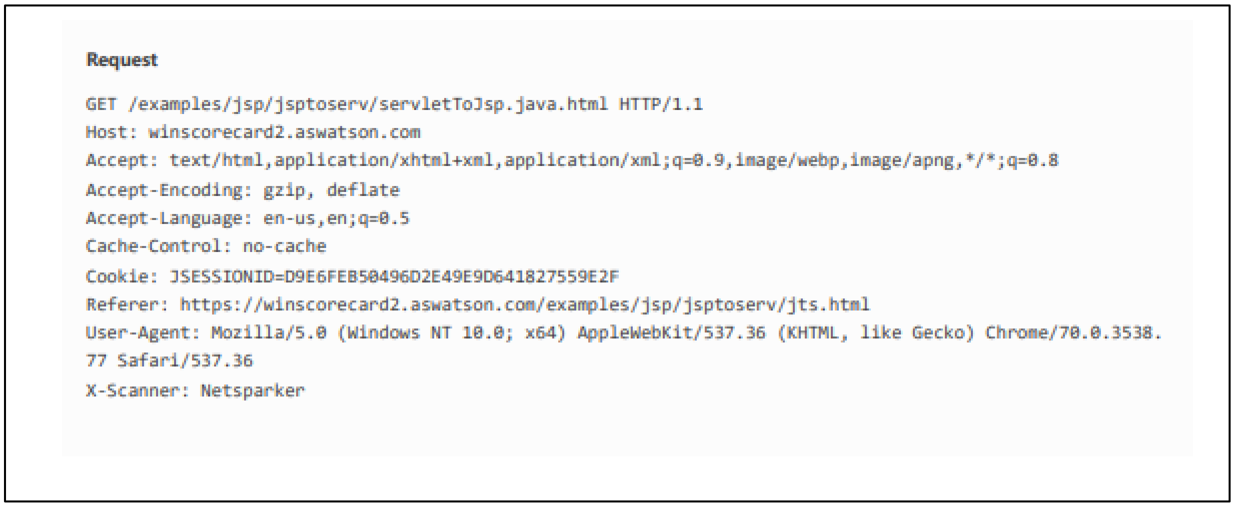
Vulnerability scanning using netsparker


Vulnerabilities

3.1. https://winscorecard2.aswatson.com/examples/jsp/jsptoserv/servletToJsp.java.html

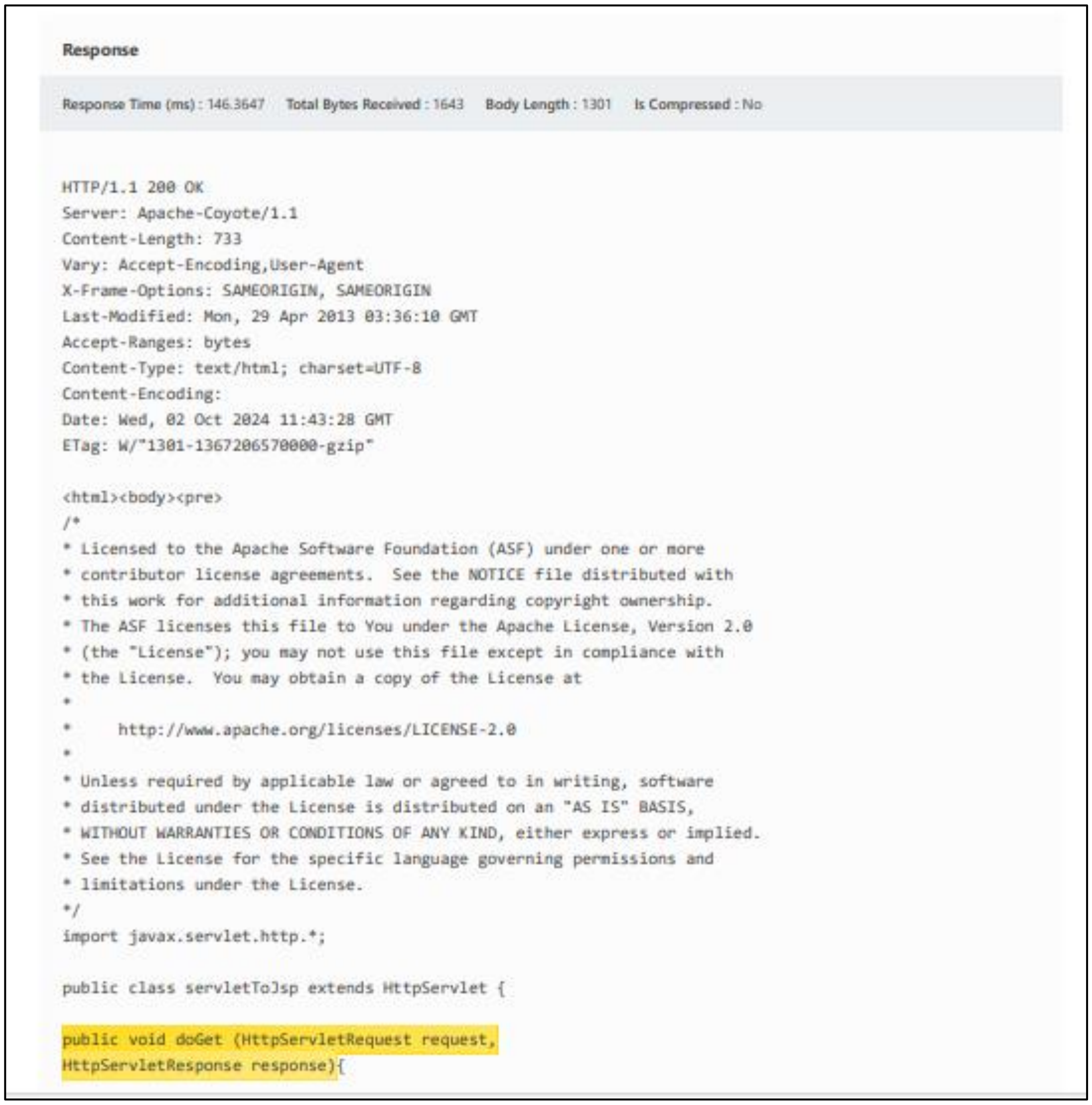
Certainty

- **Request**


Request

```
GET /examples/jsp/jsptoserv/servletToJsp.java.html HTTP/1.1
Host: winscorecard2.aswatson.com
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-us,en;q=0.5
Cache-Control: no-cache
Cookie: JSESSIONID=D9E6FEB50496D2E49E9D641827559E2F
Referer: https://winscorecard2.aswatson.com/examples/jsp/jsptoserv/jts.html
User-Agent: Mozilla/5.0 (Windows NT 10.0; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.
77 Safari/537.36
X-Scanner: Netsparker
```

- **Response**



Response

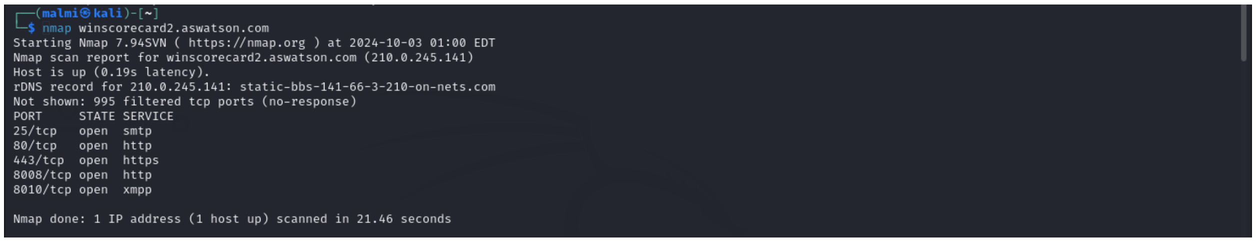
Response Time (ms) : 146.3647 Total Bytes Received : 1643 Body Length : 1301 Is Compressed : No

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: 733
Vary: Accept-Encoding,User-Agent
X-Frame-Options: SAMEORIGIN, SAMEORIGIN
Last-Modified: Mon, 29 Apr 2013 03:36:10 GMT
Accept-Ranges: bytes
Content-Type: text/html; charset=UTF-8
Content-Encoding:
Date: Wed, 02 Oct 2024 11:43:28 GMT
ETag: W/"1301-1367206570000-gzip"

<html><body><pre>
/*
* Licensed to the Apache Software Foundation (ASF) under one or more
* contributor license agreements.  See the NOTICE file distributed with
* this work for additional information regarding copyright ownership.
* The ASF licenses this file to You under the Apache License, Version 2.0
* (the "License"); you may not use this file except in compliance with
* the License.  You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
import javax.servlet.http.*;

public class servletToJsp extends HttpServlet {

public void doGet (HttpServletRequest request,
HttpServletResponse response){
```

Vulnerability finding using namp

Used command

- nmap winscorecard2.aswatson.com



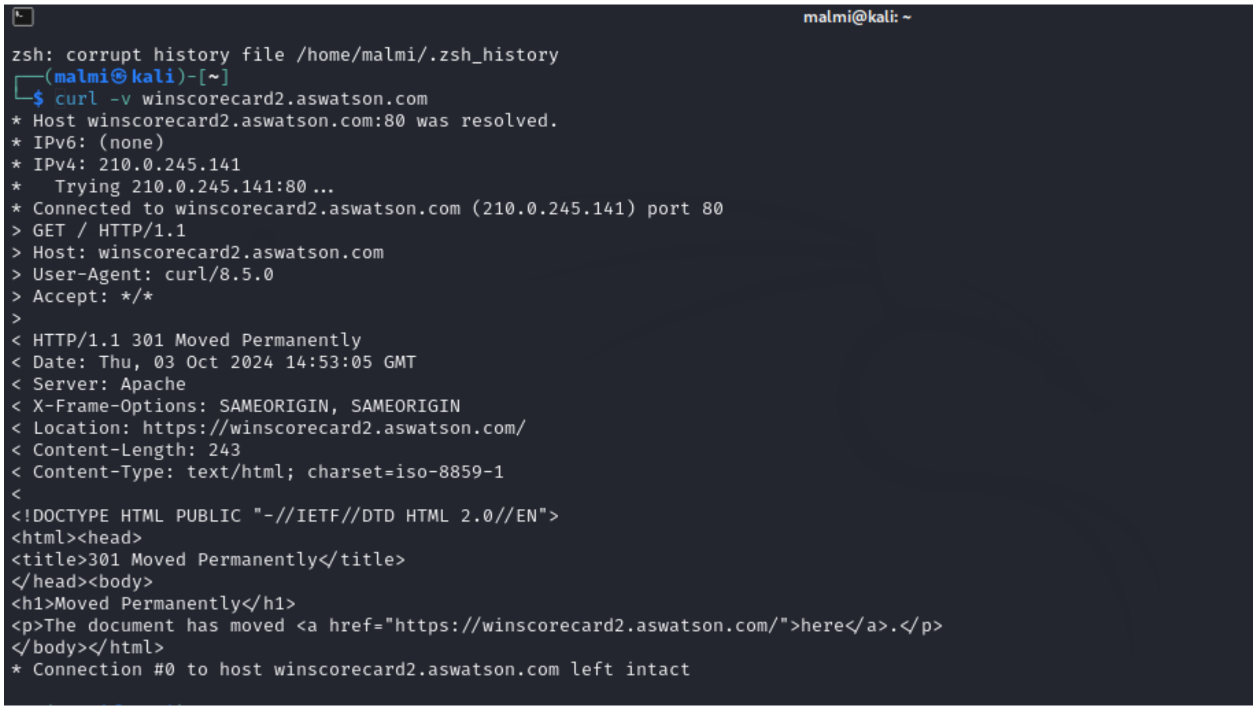
```
┌──(malmi㉿kali)-[~]
└─$ nmap winscorecard2.aswatson.com
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-03 01:00 EDT
Nmap scan report for winscorecard2.aswatson.com (210.0.245.141)
Host is up (0.19s latency).
rDNS record for 210.0.245.141: static-bbs-141-66-3-210-on-nets.com
Not shown: 995 filtered tcp ports (no-response)
PORT     STATE SERVICE
25/tcp   open  smtp
80/tcp   open  http
443/tcp  open  https
8008/tcp open  http
8010/tcp open  xmpp

Nmap done: 1 IP address (1 host up) scanned in 21.46 seconds
```

Founded open port

- 25/tcp open smtp
- 80/tcp open http
- 443/tcp open https
- 8008/tcp open http
- 8010/tcp open xmpp

Vulnerability finding using curl

- **Used command**
  - curl -v winscorecard2.aswatson.com



```
malmi@kali: ~

zsh: corrupt history file /home/malmi/.zsh_history
┌──(malmi㉿kali)-[~]
└─$ curl -v winscorecard2.aswatson.com
* Host winscorecard2.aswatson.com:80 was resolved.
* IPv6: (none)
* IPv4: 210.0.245.141
*   Trying 210.0.245.141:80...
* Connected to winscorecard2.aswatson.com (210.0.245.141) port 80
> GET / HTTP/1.1
> Host: winscorecard2.aswatson.com
> User-Agent: curl/8.5.0
> Accept: */*
>
< HTTP/1.1 301 Moved Permanently
< Date: Thu, 03 Oct 2024 14:53:05 GMT
< Server: Apache
< X-Frame-Options: SAMEORIGIN, SAMEORIGIN
< Location: https://winscorecard2.aswatson.com/
< Content-Length: 243
< Content-Type: text/html; charset=iso-8859-1
<
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="https://winscorecard2.aswatson.com/">here</a>.</p>
</body></html>
* Connection #0 to host winscorecard2.aswatson.com left intact
```

- **Results**
  - The output shows that the original request to http://winscorecard2.aswatson.com redirects to https://winscorecard2.aswatson.com/. While HTTPS is a positive step for securing connections, it doesn't inherently protect against source code exposure if those files are accessible.
  - The Content-Type: text/html; charset=iso-8859-1 header suggests the server is returning HTML content, but if sensitive files (like .java or .class) can be accessed directly

Proposed mitigation or fix

- **Secure File Permissions:**
  All sensitive files and directories, including the /WEB-INF/ directory, should have restrictive permissions applied. Only real users who need access to those directories and files should be granted access.
  Block direct access to source and configuration files on the web server configuration.

- **Proper Server Configuration:**
  Configure the web server to refuse serving access to pages of sensitive directories. Block access to the /WEB-INF/ and /META-INF/ directories using web server configuration.
  Enact rewriting of URLs or implement restrictions on path access using the web.xml file.

- **Best Practices for Application Security:**
  Scanning of the web application on a regular basis for configuration-related security vulnerabilities.
  Best practices concerning secure coding will minimize the disclosure of sensitive information.

- **Keep Software Up to Date**:
  Regularly update the application server and libraries to recent versions. It includes patching of well-known vulnerabilities, which might lead to source code disclosure.

- **Enable Security Headers:**
  Implement security headers like Content-Security-Policy and X-Content-Type-Options to mitigate several types of attacks.

- **Testing and Validation:**
  Perform regular penetration testing, finding and fixing potential issues before they could be exploited.
  Utilize tools that automatically scan for misconfigurations and other weaknesses in the web application.