



# SLIIT

---

*Discover Your Future*

---

Sri Lanka Institute of Information Technology

## Bug Bounty Report 10

wac.aswatson.com

IE2062 – Web Security

Submitted by:

**IT22227836 – ADHIKARI A.M.V.B.M**

Date of submission

2024.10.31

## Table of Contents

Report 02 – wac.aswatson.com .....	3
Vulnerability detected .....	4
Vulnerability .....	6
1. Title - Session Cookie Not Marked as Secure.....	6
Description.....	6
Affected components .....	7
Impact Assessment.....	8
Steps to reproduce.....	9
Proof of concept (if applicable).....	10
Vulnerability scanning using Netsparker.....	10
Vulnerability finding using namp.....	12
Vulnerability finding using curl .....	12
Proposed mitigation or fix .....	13
2. Title - Out-of-date Version (jQuery Validation) .....	15
Description.....	15
Impact Assessment.....	17
Steps to reproduce.....	18
Proof of concept (if applicable).....	19
Vulnerability scanning using Netsparker.....	19
Vulnerability finding using namp.....	20
Vulnerability finding using Nikto .....	21
Proposed mitigation or fix .....	22

## Report 10 – wac.aswatson.com

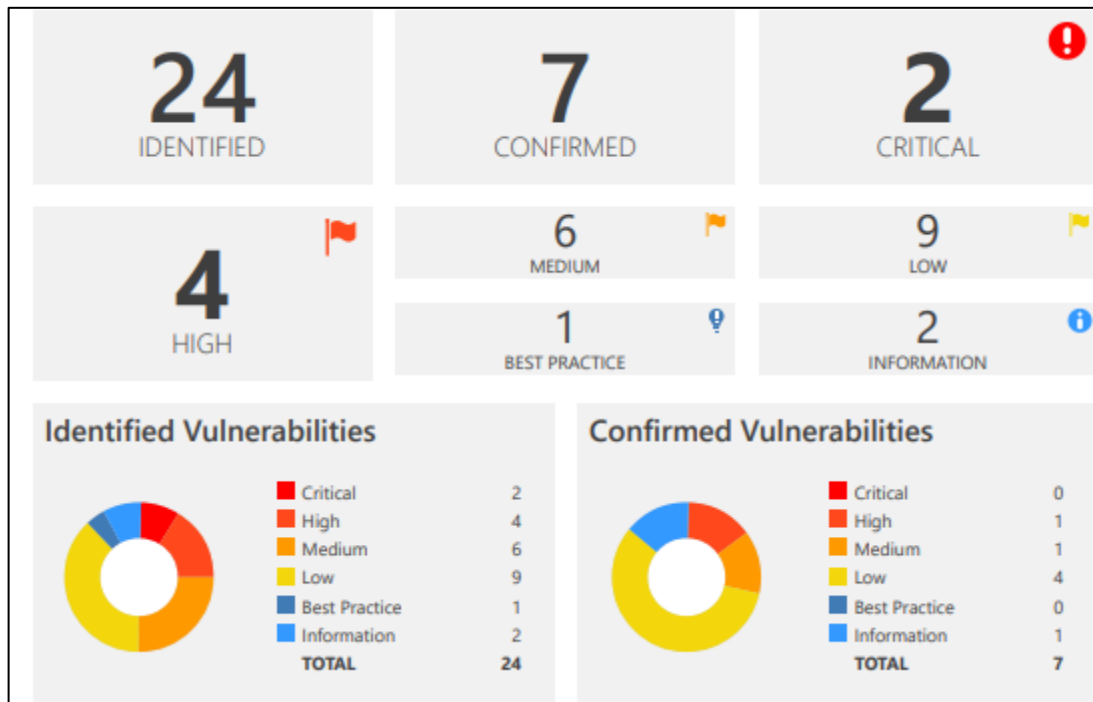
Main domain	<a href="https://www.aswatson.com/">https://www.aswatson.com/</a>
Sub domain	wac.aswatson.com
IP address	210.0.245.112
platform	HackerOne













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



Vulnerabilities By OWASP 2017				
CONFIRM	VULNERABILITY	METHOD	URL	SEVERITY
A3 - SENSITIVE DATA EXPOSURE				
1	<a href="#">Session Cookie Not Marked as Secure</a>	GET	<a href="https://wac.aswatson.com/">https://wac.aswatson.com/</a>	HIGH
1	<a href="#">Weak Ciphers Enabled</a>	GET	<a href="https://wac.aswatson.com/">https://wac.aswatson.com/</a>	MEDIUM
1	<a href="#">HTTP Strict Transport Security (HSTS) Policy Not Enabled</a>	GET	<a href="https://wac.aswatson.com/">https://wac.aswatson.com/</a>	MEDIUM
1	<a href="#">Cookie Not Marked as Secure</a>	GET	<a href="https://wac.aswatson.com/">https://wac.aswatson.com/</a>	LOW
1	<a href="#">[Possible] Internal IP Address Disclosure</a>	GET	<a href="https://wac.aswatson.com/wp-json/wp/v2/users/1">https://wac.aswatson.com/wp-json/wp/v2/users/1</a>	LOW
1	<a href="#">Referrer Policy Not Implemented</a>	GET	<a href="https://wac.aswatson.com/">https://wac.aswatson.com/</a>	BEST PRACTICE
A5 - BROKEN ACCESS CONTROL				
1	<a href="#">JIS Store File Found</a>	GET	<a href="https://wac.aswatson.com/DS_Store">https://wac.aswatson.com/DS_Store</a>	LOW
A6 - SECURITY MISCONFIGURATION				
1	<a href="#">Cookie Not Marked as HttpOnly</a>	GET	<a href="https://wac.aswatson.com/">https://wac.aswatson.com/</a>	LOW
1	<a href="#">Insecure Frame (External)</a>	GET	<a href="https://wac.aswatson.com/">https://wac.aswatson.com/</a>	LOW
1	<a href="#">Windows Short Filenames</a>	OPTIONS	<a href="https://wac.aswatson.com/wp-content/plugins/*-1*%5ca.aspx/as.psemorpath+/">https://wac.aswatson.com/wp-content/plugins/*-1*%5ca.aspx/as.psemorpath+/</a>	LOW
1	<a href="#">[Possible] Phishing by Navigating Browser Tabs</a>	GET	<a href="https://wac.aswatson.com/">https://wac.aswatson.com/</a>	LOW
1	<a href="#">Missing X-Frame-Options Header</a>	GET	<a href="https://wac.aswatson.com/">https://wac.aswatson.com/</a>	LOW
1	<a href="#">Version Disclosure (PHP)</a>	GET	<a href="https://wac.aswatson.com/">https://wac.aswatson.com/</a>	LOW
1	<a href="#">OPTIONS Method Enabled</a>	OPTIONS	<a href="https://wac.aswatson.com/wp-includes/">https://wac.aswatson.com/wp-includes/</a>	INFORMATION

CONFIRM	VULNERABILITY	METHOD	URL	SEVERITY
	<a href="#">.htaccess File Detected</a>	GET	<a href="https://wac.aswatson.com/.htaccess">https://wac.aswatson.com/.htaccess</a>	<span>INFORMATION</span>
<b>A9 - USING COMPONENTS WITH KNOWN VULNERABILITIES</b>				
	<a href="#">Out-of-date Version (PHP)</a>	GET	<a href="https://wac.aswatson.com/">https://wac.aswatson.com/</a>	<span>CRITICAL</span>
	<a href="#">Out-of-date Version (WordPress)</a>	GET	<a href="https://wac.aswatson.com/wp-includes/images/arrow-pointer-blue.png">https://wac.aswatson.com/wp-includes/images/arrow-pointer-blue.png</a>	<span>CRITICAL</span>
	<a href="#">Out-of-date Version (GSAP)</a>	GET	<a href="https://wac.aswatson.com/wp-content/plugins/essential-grid/public/assets/js/jquery.themepunch.tools.min.js">https://wac.aswatson.com/wp-content/plugins/essential-grid/public/assets/js/jquery.themepunch.tools.min.js</a>	<span>HIGH</span>
	<a href="#">Out-of-date Version (jQuery Validation)</a>	GET	<a href="https://wac.aswatson.com/registration/vendor/jquery-validation/dist/additional-methods.min.js">https://wac.aswatson.com/registration/vendor/jquery-validation/dist/additional-methods.min.js</a>	<span>HIGH</span>
	<a href="#">Out-of-date Version (Moment.js)</a>	GET	<a href="https://wac.aswatson.com/zh/event/hong-kong-junior-age-group-athletics-competition-2024-5/">https://wac.aswatson.com/zh/event/hong-kong-junior-age-group-athletics-competition-2024-5/</a>	<span>HIGH</span>
	<a href="#">[Possible] BREACH Attack Detected</a>	GET	<a href="https://wac.aswatson.com/zh/events/?_wp_http_referer=%2fevents%2Flist%2Fsubmit-bar=3&amp;tribe-events-views%5B_wpnonce%5d=24cb96372a&amp;tribe-events-views%5Btribe-bar-search%5d=3&amp;tribe-events-views%5Burl%5d=https%3A%2F%2Fwac.aswatson.com%2Fevents%2Flist%2F">https://wac.aswatson.com/zh/events/?_wp_http_referer=%2fevents%2Flist%2Fsubmit-bar=3&amp;tribe-events-views%5B_wpnonce%5d=24cb96372a&amp;tribe-events-views%5Btribe-bar-search%5d=3&amp;tribe-events-views%5Burl%5d=https%3A%2F%2Fwac.aswatson.com%2Fevents%2Flist%2F</a>	<span>MEDIUM</span>
	<a href="#">Out-of-date Version (Bootstrap)</a>	GET	<a href="https://wac.aswatson.com/wac-introduction/">https://wac.aswatson.com/wac-introduction/</a>	<span>MEDIUM</span>
	<a href="#">Out-of-date Version (jQuery UI Dialog)</a>	GET	<a href="https://wac.aswatson.com/wp-content/plugins/wac-enrollment-form/cjs/jquery-ui/jquery-ui.js">https://wac.aswatson.com/wp-content/plugins/wac-enrollment-form/cjs/jquery-ui/jquery-ui.js</a>	<span>MEDIUM</span>
	<a href="#">Out-of-date Version (jQuery)</a>	GET	<a href="https://wac.aswatson.com/">https://wac.aswatson.com/</a>	<span>MEDIUM</span>

## Vulnerability

### 1. Title - Session Cookie Not Marked as Secure

#### 5. Session Cookie Not Marked as Secure

HIGH  1

CONFIRMED  1

Netsparker identified a session cookie not marked as secure, and transmitted over HTTPS.

This means the cookie could potentially be stolen by an attacker who can successfully intercept the traffic, following a successful man-in-the-middle attack.

It is important to note that Netsparker inferred from the its name that the cookie in question is session related.

##### Impact

This cookie will be transmitted over a HTTP connection, therefore an attacker might intercept it and hijack a victim's session. If the attacker can carry out a man-in-the-middle attack, he/she can force the victim to make an HTTP request to your website in order to steal the cookie.

❖ Risk – High

## Description

Session Cookie Not Marked as Secure: This vulnerability is related to those session cookies being set without the "Secure" attribute, although they are transmitted over an HTTPS connection. It means that, in case an attacker lures a user into performing any action over HTTP instead of HTTPS, such an exposed cookie could be intercepted in a man-in-the-middle attack. Such a vulnerability is supposed to be very dangerous if the users operate in an environment that will expose them to connecting through public or unsecured networks because an attacker can intercept the network traffic and steal the session cookies.

Once an attacker gets access to the session cookie, he can hijack the user's session and may gain unauthorized access to sensitive information or impersonate the victim within the application. This could consequently lead to information leaks and unauthorized activities, and even privilege escalation when the hijacked session happens to be owned by a user with privileges greater than normal, like an administrator. To mitigate this vulnerability, all session-related cookies need to have the Secure attribute set. This ensures such data will only be transmitted over encrypted HTTPS; hence, it will be prevented from becoming exposed in insecure contexts.

## Affected components

Affected components due to this vulnerability because of which the session cookie is not marked as secure:

- **Web Application Session Management:**

Of all, the most significant component affected would be the session management system of the web application. More precisely, this attack vector affects cookies related to sessions or any others that manage authentication data along with session data.

- **Cookies and HTTP Headers:**

Based on vulnerability, cookies are set and sent by HTTP headers. The "Secure" attribute not being present in the response header of HTTP is one major misconfiguration.

- **Network Layer:**

If the cookies are not flagged as secure, they might be transmitted over insecure HTTP connections if the web application is served over HTTPS. This exposes the cookies to network-layer attacks, including man-in-the-middle attacks.

- **User Sessions:**

The hijacking of user sessions is possible, allowing unauthorized access to sensitive information or system resources. The application would be disclosing the Secure attribute's existence and weakening its session management and general security position. This kind of condition can be so easily exploited by attackers.

## Impact Assessment

The vulnerability for not setting the secure flag of the session cookie can be quite substantial in its impact, since users' sessions might get compromised due to:

- **Session Hijacking:**  
Because the session cookie may be sent on an insecure channel, it will be possible for an attacker to sniff that cookie by performing a man-in-the-middle attack. Once the session cookie is intercepted, the attacker will further be able to impersonate the legitimate user, which gives him unauthorized access to the user session and data.
- **Data Theft:**  
The compromised session might reveal the divulging of sensitive information, such as personal data, transactional history, or other important data getting accessed or stored during that session.
- **Privacy Violation:**  
This may also involve intercepting the session cookie to track a user's actions or monitor their online activities, which is essentially a violation of their privacy.
- **Compliance Risk:**  
Failure to secure session cookies will be a violation of the security standards like PCI DSS and OWASP, which require that session cookies should be secured. This may lead to some legal penalties or loss of certifications.

The vulnerability poses a medium to high risk regarding user privacy and system security when considered in conjunction with other attack vectors such as MITM.



## Steps to reproduce

- **Open a Browser:**  
Use any browser or an intercepting proxy-for example, Burp Suite or OWASP ZAP-to capture HTTP/HTTPS traffic.
- **Access the Target Website:**  
Open a browser and navigate to <https://wac.aswatson.com>. If requested, log in or otherwise induce the web application into creating a session.
- **Inspect HTTP Cookies:**
  - Using the browser developer tools-in Chrome or Firefox, press F12 to open the developer tools-navigate to the "Network" tab, and review response headers or application cookies.
  - Now look for PHPSESSID cookie, which is the session cookie.
- **Check Cookie Attributes:**  
From the attributes, check whether the Secure flag is present in that cookie. If the cookie should be sent only over HTTPS, the secure flag shall be there. Otherwise, the session cookie is vulnerable to an interception attack.
- **Perform a MITM Attack:**
  - Employ a man-in-the-middle attack using a utility such as Wireshark or Ettercap on a controlled environment.
  - Analyze the HTTP traffic and see whether the session cookie PHPSESSID is sent unencrypted.
- **Confirm Vulnerability:**  
If the session cookie is sent without the Secure flag, it confirms the vulnerability because, on this unsecured environment, it can be sniffed.

## Proof of concept (if applicable)

### Vulnerability scanning using Netsparker

#### Vulnerabilities

5.1. <https://wac.aswatson.com/>

**CONFIRMED**

- Request

#### Request

```
GET / HTTP/1.1
Host: wac.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): 1706.646    Total Bytes Received : 95709    Body Length : 94861    Is Compressed : No

```
HTTP/1.1 200 OK
Cache-Control: no-store, no-cache, must-revalidate
X-TEC-API-ROOT: https://wac.aswatson.com/wp-json/tribe/events/v1/
Set-Cookie: PHPSESSID=m8o77fgnai7bs9bmsgmp872jt8; path=/
Set-Cookie: qtrans_front_language=en; expires=Wed, 08-Oct-2025 15:53:50 GMT; Max-Age=31536000; path=/
X-Powered-By: PHP/7.3.20
Server: Apache
Link: <https://wac.aswatson.com/wp-json/>; rel="https://api.w.org/", <https://wac.aswatson.com/>; rel=shortlink
X-Content-Type-Options: nosniff
X-TEC-API-VERSION: v1
X-XSS-Protection: 1; mode=block
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Vary: Accept-Encoding,User-Agent
Content-Length: 20764
X-Pingback: https://wac.aswatson.com/xmlrpc.php
Content-Type: text/html; charset=UTF-8
X-TEC-API-ORIGIN: https://wac.aswatson.com
Pragma: no-cache
Date: Tue, 08 Oct 2024 15:5HTTP/1.1 200 OK
Cache-Control: no-store, no-cache, must-revalidate
X-TEC-API-ROOT: https://wac.aswatson.com/wp-json/tribe/events/v1/
Set-Cookie: PHPSESSID=m8o77fgnai7bs9bmsgmp872jt8; path=/

Set-Cookie: qtrans_front_language=en; expires=Wed, 08-Oct-2025 15:53:50 GMT; Max-Age=31536000; path=/
X-Powered-By: PHP/7.3.20
Server: Apache
Link: <https://wac.aswatson.com/wp-json/>; rel="https:
```

## Vulnerability finding using nmap

### Used command

- `nmap -sV wac.aswatson.com`

```
(malmi@kali)~[~/Desktop]
$ nmap -sV wac.aswatson.com
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-08 17:26 EDT
Nmap scan report for wac.aswatson.com (210.0.245.112)
Host is up (0.100s latency).
rDNS record for 210.0.245.112: static-bbs-112-66-3-210-on-nets.com
Not shown: 997 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
25/tcp    open  smtp?
80/tcp    open  http    Apache httpd
443/tcp   open  ssl/http Apache httpd
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port25-TCP:V=7.94SVN%I=7%D=10/8%Time=6705A3CA%P=x86_64-pc-linux-gnu%r(H
SF:ello,1F,"452\x20syntax\x20error\x20(\connecting)\r\n")%r(Help,1F,"452\
SF:x20syntax\x20error\x20(\connecting)\r\n")%r(GenericLines,34,"452\x20sy
SF:ntax\x20error\x20(\connecting)\r\n421\x20too\x20many\x20errors\r\n")%r
SF:(GetRequest,34,"452\x20syntax\x20error\x20(\connecting)\r\n421\x20too\
SF:x20many\x20errors\r\n")%r(HTTPOptions,34,"452\x20syntax\x20error\x20(\c
SF:connecting)\r\n421\x20too\x20many\x20errors\r\n")%r(RTSPRequest,34,"452
SF:\x20syntax\x20error\x20(\connecting)\r\n421\x20too\x20many\x20errors\r
SF:\n");
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 86.60 seconds
```

### Founded open port

- 25/tcp open smtp?
- 80/tcp open http Apache httpd
- 443/tcp open ssl/http Apache httpd

## Vulnerability finding using curl

### Used command

- `curl -I https://wac.aswatson.com`

```
(malmi@kali)~[~]
$ curl -I https://wac.aswatson.com
HTTP/1.1 200 OK
Date: Tue, 08 Oct 2024 19:36:42 GMT
Server: Apache
X-Powered-By: PHP/7.3.20
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
X-Pingback: https://wac.aswatson.com/xmlrpc.php
Link: <https://wac.aswatson.com/wp-json/>; rel="https://api.w.org/", <https://wac.aswatson.com/>; rel=shortlink
X-TEC-API-VERSION: v1
X-TEC-API-ROOT: https://wac.aswatson.com/wp-json/tribe/events/v1/
X-TEC-API-ORIGIN: https://wac.aswatson.com
Set-Cookie: PHPSESSID=5sup35ki780pss17s53dujn1hg; path=/
Set-Cookie: qtrans_front_language=en; expires=Wed, 08-Oct-2025 19:36:42 GMT; Max-Age=31536000; path=/
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Vary: User-Agent
Content-Type: text/html; charset=UTF-8
```

## Result

- Below is the relevant portion of the curl command output.  
-Set-Cookie: PHPSESSID=5sup35ki780pss17s53dujn1hg; path=/
  - The absence of the Secure attribute next to Set-Cookie indicates that the cookie can be transmitted over both HTTP and HTTPS.

## Proposed mitigation or fix

### Actions to Take

1. See the remedy for solution.
2. Mark all cookies used within the application as secure. *(If the cookie is not related to authentication or does not carry any personal information, you do not have to mark it as secure.)*

### Remedy

Mark all cookies used within the application as secure.

### Required Skills for Successful Exploitation

To exploit this issue, the attacker needs to be able to intercept traffic. This generally requires local access to the web server or to the victim's network. Attackers need to understand layer 2 and have gained access to a system between the victim and the web server.

To minimize the vulnerability of session cookies not having the secure flag set, perform the following:

- **Set Secure Flag:**  
Ensure that all session cookies have to be marked with the "Secure" attribute. It tells the browser to send the cookie only in case of HTTPS, thus preventing interception during transmission.
- **Set the HttpOnly Flag:**  
Besides the Secure flag, set the "HttpOnly" attribute of cookies. This will prevent client-side scripts from accessing the cookie and further reduce the risk of XSS attacks.
- **Strict Transport Security:**  
Implement HTTP Strict Transport Security to force the use of secure connections. This will prevent any attempts at connecting over HTTP; therefore, all traffic will be encrypted.

- **Perform Cookie Auditing:**  
Periodically perform cookie audits on cookies utilized by an application. Enumerate those cookies that do not have the Secure flag set and review them for necessity.
- **Developing Education:**  
Assist development teams with supporting the need for cookie security, and best practices become part of coding standards.

By taking these measures, session hijacking can be minimized, and the overall security of the application can be greatly improved.

## 2. Tittle - Out-of-date Version (jQuery Validation)

### 4. Out-of-date Version (jQuery Validation)

HIGH  1

Netsparker identified that the target web site is using jQuery Validation and detected that it is out of date.

#### Impact

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

#### jQuery Validation Other Vulnerability

The jQuery Validation Plugin (jquery-validation) provides drop-in validation for forms. Versions of jquery-validation prior to 1.19.5 are vulnerable to regular expression denial of service (ReDoS) when an attacker is able to supply arbitrary input to the url2 method. This is due to an incomplete fix for CVE-2021-43306. Users should upgrade to version 1.19.5 to receive a patch.

❖ Risk - High

## Description

The vulnerability in question exists because of the usage of the jQuery Validation library on the target website, which is at an older version, 1.17.0, where unpatched security vulnerabilities exist. This library is quite popular and usually used for form validation. The vulnerabilities in that library can be exploited-especially a ReDoS attack. Such an attacker could produce crafted inputs intended to cause excessive resource utilization on the server in question by the affected validation methods. Eventually, this will lead to performance degradation, crashes, or even a full denial of service, which disrupts the normal functionalities of the application.

Also, lack of updates means only that the site remains vulnerable with other, earlier versions of the library under various documented CVE's. These weaknesses may expose the website to critical vulnerability concerning data integrity and confidentiality and may become a severe violation of user trust and compliance with security standards. Hence, upgrading to the latest version helps mitigate these kinds of vulnerabilities, ensuring robust security of the web application.

## Affected components

The applied scenario in this case affects mainly the following components:

- **jQuery Validation Library:**  
The component introduced in this vulnerability is the jQuery Validation plugin. It was identified by the scanning as version 1.17.0. This library is quite relevant to deal with client-side form validation. For example, user input can be checked against a specific format and rule before sending it to the server.
- **JavaScript Files:**
  - This vulnerability is specifically situated in the JavaScript file at the address:  
**<https://wac.aswatson.com/registration/vendor/jquery-validation/dist/jquery.validate.min.js>**.
  - This minified JavaScript contains the functionality of jQuery Validation, an outdated version of which may pose a high risk for exploitation.
- **Web Application Framework:** Since jQuery Validation could be bound inside web applications using different frameworks-say, for example, using PHP or Ruby on Rails-any web application that uses this version is at risk.
- **Dependent Libraries or Components:** Other components or libraries that may interact with or depend on jQuery Validation can also be indirectly affected, provided there is dependency on its functionality in terms of form processing and validation.
- **User Interfaces:** Any web pages or forms dependent on the obsolete jQuery Validation library may also be affected, since they will carry security vulnerabilities related to the handling of user input.

It exposes the entire web application, including users, to security vulnerabilities from the outdated version of jQuery Validation, mostly in form submissions and input validation.



## Impact Assessment

Impact assessment related to the vulnerability could be registered as follows for an outdated jQuery Validation library, version 1.17.0:

- **Security Risks:**  
The possibility of Regular Expression Denial of Service (ReDoS) attacks is the main source of risk. Attackers can transmit specially constructed inputs that cause high resource usage in order to take advantage of the vulnerabilities in the out-of-date library, especially when using the url2 method. Because of the server's difficulty handling malicious requests, this could result in service interruptions or deterioration.
- **Data Integrity:**  
Assuming the bugs allow bypassing input validation, an attacker may input some premeditated information, malicious scripts, or malware, which will result in a Cross-Site Scripting (XSS) attack. Therefore, it is not only about the integrity compromise of user data but also risks the exposure of sensitive information.
- **User Trust and Experience:**  
Security vulnerabilities can shatter user trusts. If users get wind that their data might surface or the application is not secure, they will probably avoid using the site, which means loss of reputation and users. Furthermore, the experience can be destroyed when the application slows down or becomes unresponsive due to exploitation.
- **Regulatory Compliance:**  
Most industries have imposed regulations, such as PCI DSS, HIPAA, etc., that necessitate the installation of updated software so that sensitive information is protected. If this vulnerability remains unpatched, it could mean failure of compliance, which can result in some legal consequences and fines.
- **Long-term Implications:**  
If left unattended, the aggregation of vulnerabilities within deprecated components can provide a wider attack surface for adversaries, thus giving way to more critical breaches or exploitations later on.

This means that the old version of jQuery Validation brings not only instantaneous problems in performance and data integrity, but also long-term effects concerning trust by users and compliance with standards of security. Organizations are strongly encouraged to upgrade to the latest stable version to mitigate these risks.

## Steps to reproduce

- **Log in to Website:**  
Open any Web browser and go to <https://wac.aswatson.com/registration/vendor/jquery-validation/dist/jquery.validate.min.js>.
- **Check Version:**  
Look into response headers or the body, check the version. One can notice that the detected version is 1.17.0.
- **Verify with Latest Version:**  
Go to the official jQuery Validation site or repository (say, GitHub) and check the latest stable version; this is 1.21.0.
- **Known Vulnerabilities Identification:**  
Using the National Vulnerability Database or the Common Vulnerabilities and Exposures database, research using any available resources known vulnerabilities associated with the identified version, 1.17.0. The search should be done based on ReDoS vulnerabilities.
- **Input Simulation:**  
Using the application, if possible, simulate an event in which an attacker would inject malicious or large payloads that may expose the ReDoS vulnerability.
- **Observe Application Behavior:**  
Observe response time and behavior of the application. Slow or unresponsive responses by the application will show that the ReDoS has been exploited successfully.

## Proof of concept (if applicable)

### Vulnerability scanning using Netsparker

**External References**

- [CVE-2021-21252](#)

**Vulnerabilities**

4.1. <https://wac.aswatson.com/registration/vendor/jquery-validation/dist/additional-methods.min.js>

36 / 106

**Identified Version**

- 1.17.0

**Latest Version**

- 1.21.0 (in this branch)

**Vulnerability Database**

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

**Certainty**

#### ▪ Request

**Request**

```
GET /registration/vendor/jquery-validation/dist/additional-methods.min.js HTTP/1.1
Host: wac.aswatson.com
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-us,en;q=0.5
Cache-Control: no-cache
Cookie: PHPSESSID=v5jff8id3765okn12qnqs6co8u7; qtrans_front_language=zh
Referer: https://wac.aswatson.com/registration/index.php?lang=zh-HK
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) : 1397.9578    Total Bytes Received : 18810    Body Length : 18464    Is Compressed : No

```
HTTP/1.1 200 OK
Server: Apache
X-Content-Type-Options: nosniff
Vary: Accept-Encoding,User-Agent
X-XSS-Protection: 1; mode=block
Content-Length: 5457
Last-Modified: Mon, 17 Apr 2023 04:40:06 GMT
Accept-Ranges: bytes
Content-Type: text/javascript
Content-Encoding:
Date: Tue, 08 Oct 2024 17:13:50 GMT
ETag: "4820-5f98

-
: 5457
Last-Modified: Mon, 17 Apr 2023 04:40:06 GMT
Accept-Ranges: bytes
Content-Type: text/javascript
Content-Encoding:
Date: Tue, 08 Oct 2024 17:13:50 GMT
ETag: "4820-5f980c8f23580-gzip"

/*! jQuery Validation Plugin - v1.17.0- 7/29/2017
* https://jqueryvalidation.org/
* Copyright (c) 2017 Jörn Zaefferer; Licensed MIT */
!function(a){"function"==typeof define&&define.amd?define(["jquery","./jquery.validate.min"],a):"obj
-
```

## Vulnerability finding using nmap

### Used command

- `nmap -sV wac.aswatson.com`

```
(malini@kali) ~[~/Desktop]
$ nmap -sV wac.aswatson.com
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-08 17:26 EDT
Nmap scan report for wac.aswatson.com (210.0.245.112)
Host is up (0.100s latency).
rDNS record for 210.0.245.112: static-bbs-112-66-3-210-on-nets.com
Not shown: 997 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
25/tcp    open  smtp?
80/tcp    open  http   Apache httpd
443/tcp   open  ssl/http Apache httpd
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port25-TCP:V=7.94SVN|I=7%D=10/8%Time=6705A3CA|P=x86_64-pc-linux-gnu|(H
SF:ello,1F,"452\x20syntax\x20error\x20(\connecting)\r\n")%(Help,1F,"452\
SF:x20syntax\x20error\x20(\connecting)\r\n")%(GenericLines,34,"452\x20sy
SF:ntax\x20error\x20(\connecting)\r\n421\x20too\x20many\x20errors\r\n")%(
SF:(GetRequest,34,"452\x20syntax\x20error\x20(\connecting)\r\n421\x20too\
SF:x20many\x20errors\r\n")%(HTTPOptions,34,"452\x20syntax\x20error\x20(\c
SF:connecting)\r\n421\x20too\x20many\x20errors\r\n")%(RTSPRequest,34,"452
SF:x20syntax\x20error\x20(\connecting)\r\n421\x20too\x20many\x20errors\r
SF:n");
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 86.60 seconds
```

## Founded open port

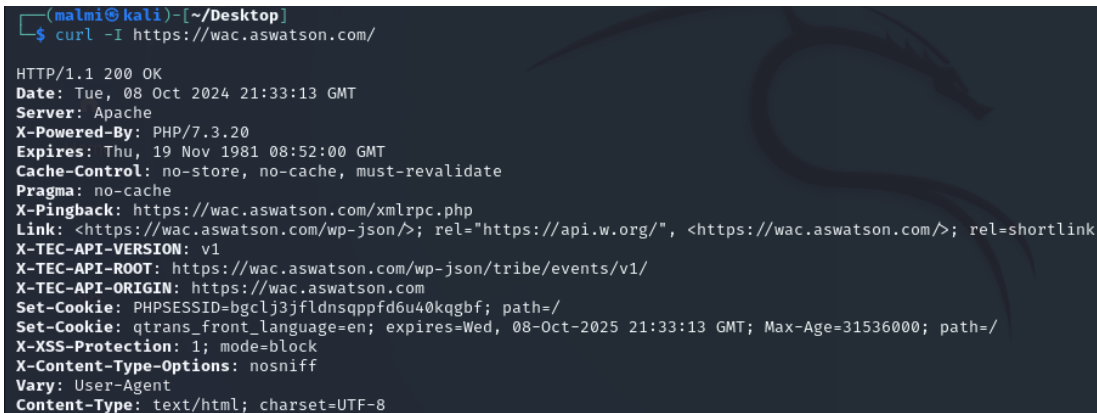
- 25/tcp open smtp?
- 80/tcp open http Apache httpd
- 443/tcp open ssl/http Apache httpd

The Nmap scan results show that the application is running, and further checks indicate that jQuery Validation version 1.17.0 is in use.

## Vulnerability finding using Nikto

### ■ Used command

- `curl -I https://wac.aswatson.com/`



```
(malini@kali) - [~/Desktop]
$ curl -I https://wac.aswatson.com/

HTTP/1.1 200 OK
Date: Tue, 08 Oct 2024 21:33:13 GMT
Server: Apache
X-Powered-By: PHP/7.3.20
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
X-Pingback: https://wac.aswatson.com/xmlrpc.php
Link: <https://wac.aswatson.com/wp-json/>; rel="https://api.w.org/", <https://wac.aswatson.com/>; rel=shortlink
X-TEC-API-VERSION: v1
X-TEC-API-ROOT: https://wac.aswatson.com/wp-json/tribe/events/v1/
X-TEC-API-ORIGIN: https://wac.aswatson.com
Set-Cookie: PHPSESSID=bgclj3jfldnsqppfd6u40kqgbf; path=/
Set-Cookie: qtrans_front_language=en; expires=Wed, 08-Oct-2025 21:33:13 GMT; Max-Age=31536000; path=/
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Vary: User-Agent
Content-Type: text/html; charset=UTF-8
```

### ■ Results

Based on the scan results, the following information has been extracted:

- Vulnerable Component: jQuery Validation
- Identified Version: 1.17.0
- Latest Version: 1.21.0
- Vulnerability Type: Outdated library version with known vulnerabilities.
- The vulnerability scanning tool has identified this outdated version, indicating a medium risk according to CVSS scoring

## Proposed mitigation or fix

To mitigate this vulnerability with the jQuery Validation version, which is out of date at 1.17.0, on the web application <https://wac.aswatson.com/>, the following can be done:

- **Update jQuery Validation Plugin:**  
The jQuery Validation plugin needs to be updated to at least the most recent stable version, 1.21.0 or newer. This version comes with considerable patches in security and improvements for fixing vulnerabilities, including issues related to ReDoS.
- **Periodic Security Audits:**  
Create a schedule of regular security audits and/or scans to identify outdated libraries, hence vulnerabilities. Tools that can be used for detecting outdated dependencies include OWASP Dependency-Check, Snyk, or npm audit.
- **Implement Content Security Policy (CSP):**  
Introduce Content Security Policy, which will restrict the sources from where scripts will be allowed to load. This might help reduce the risk of working with outdated libraries.
- **Using Package Managers:**  
Consider using a package manager, like npm, that does provide the functionality of dependency management and automatically notifying about versions that have gone out of date to ensure timely updates.
- **Review and Test After Updates:**  
Once the plugin has been updated, extensive testing of the web application should be done to ensure all functionalities behave as they should without this update introducing a new set of problems.
- **Monitor Vulnerability Databases:**  
Subscribe to various vulnerability databases and security advisories around jQuery and other libraries. Sources could include: NVD, CVE, and/or notification by the vendor.
- **Development Team Education:**  
Arrange for training in secure coding practices of the development team, including how to keep libraries and frameworks current.

Performing these mitigations will go a long way in reducing the risk posed by the now-outdated jQuery Validation plugin and ensure security and stability for the web application as a whole.