WEB APPLICATION PENETRATION TESTING REPORT

Conducted by

Temitope Abidemi
PenTester/Soc Analyst

royalcysec@gmail.com
+2347036876391

2023-01-18

WEB APPLICATION PENETRATION TESTING

General introduction

Website/server vulnerabilities are weaknesses or flaws in a website or web application that can be exploited by attackers to gain unauthorized access, steal sensitive information, or perform other malicious actions. These vulnerabilities can exist in the website's code, server configuration, or third-party software that is used by the website. Some common types of website vulnerabilities include SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF). It's important for website owners and developers to regularly scan and test their websites for vulnerabilities in order to identify and fix them before they can be exploited by attackers.

After using some tools I discovered the vulnerabilities on the following given URL:

- http://rest.vulnweb.com/ (REST/OPEN API)
- http://testphp.vulnweb.com/ (Website)
- http://testhtml5.vulnweb.com/ (Website)
- http://testaspnet.vulnweb.com/ (Website)

http://rest.vulnweb.com/ ----- [200 OK] Apache[2.4.25], Country[UNITED STATES][US], HTML5, HTTPServer[Debian Linux][Apache/2.4.25 (Debian)], IP[35.81.188.86], Meta-Author[Acunetix], PHP[7.1.26], Title[Acunetix Vulnerable REST API], X-Powered-By[PHP/7.1.26]

http://testphp.vulnweb.com/ ---- [200 OK] ActiveX[D27CDB6E-AE6D-11cf-96B8-444553540000], Adobe-Flash, Country[UNITED STATES][US], Email[wvs@acunetix.com], HTTPServer[nginx/1.19.0], IP[44.228.249.3], Object[http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,29,0][clsid:D27CDB6E-AE6D-11cf-96B8-444553540000], PHP[5.6.40-38+ubuntu20.04.1+deb.sury.org+1], Script[text/JavaScript], Title[Home of Acunetix Art], X-Powered-By[PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1], nginx[1.19.0]

http://testhtml5.vulnweb.com/ ---- [200 OK] Bootstrap[2.3.1], Country[UNITED STATES][US], HTML5, HTTPServer[nginx/1.19.0], IP[44.228.249.3], JQuery[1.9.1], PasswordField[password], Script, Title[SecurityTweets - HTML5 test website for Acunetix Web Vulnerability Scanner], UncommonHeaders[access-control-allow-origin], nginx[1.19.0]

http://testaspnet.vulnweb.com/ ---- [200 OK] ASP_NET[2.0.50727], Cookies[ASP.NET_SessionId], Country[UNITED STATES][US], HTTPServer[Microsoft-IIS/8.5], HttpOnly[ASP.NET_SessionId], IP[44.238.29.244], MetaGenerator[Microsoft Visual Studio .NET 7.1], Microsoft-IIS[8.5], Title[acublog news], X-Powered-By[ASP.NET]

Observations

Then after going deep on the research of my quest in finding the websites/server vulnerabilities; I discovered they have some things in common and later concluded on to dive deep on the 2nd URL [http://testphp.vulnweb.com/].

This report will be based only on [http://testphp.vulnweb.com/].

Broken Cryptography

(Password Transmitted over HTTP, SSL/TLS Not Implemented)

Target: http://testphp.vulnweb.com/

Technical severity: High

Vulnerability details:

- http://testphp.vulnweb.com/login.php

Description

The application allows users to connect to it over unencrypted connections. An attacker suitably positioned to view a legitimate user's network traffic could record and monitor their interactions with the application and obtain any information the user supplies. Furthermore, an attacker able to modify traffic could use the application as a platform for attacks against its users and third-party websites. Unencrypted connections have been exploited by ISPs and governments to track users, and to inject adverts and malicious JavaScript. Due to these concerns, web browser vendors are planning to visually flag unencrypted connections as hazardous.

To exploit this vulnerability, an attacker must be suitably positioned to eavesdrop on the victim's network traffic. This scenario typically occurs when a client communicates with the server over an insecure connection such as public Wi-Fi, or a corporate or home network that is shared with a compromised computer. Common defenses such as switched networks are not sufficient to prevent this. An attacker situated in the user's ISP or the application's hosting infrastructure could also perform this attack. Note that an advanced adversary could potentially target any connection made over the Internet's core infrastructure.

Please note that using a mixture of encrypted and unencrypted communications is an ineffective defense against active attackers, because they can easily remove references to encrypted resources when these references are transmitted over an unencrypted connection.

It was detected that password data is being transmitted over HTTP.

Solution

Applications should use transport-level encryption (SSL/TLS) to protect all communications passing between the client and the server. The Strict-Transport-Security HTTP header should be used to ensure that clients refuse to access the server over an insecure connection.

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.

Server-Side Injection

(SQL Injection, Local File Inclusion, Using Default Credentials)

Target: http://testphp.vulnweb.com/

Technical severity: high

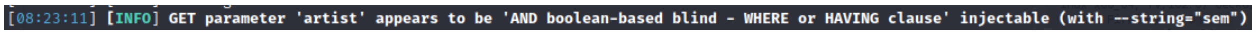
Vulnerability details

- http://testphp.vulnweb.com/listproducts.php?artist=1%20OR%2017-7%3d10
- http://testphp.vulnweb.com/Mod_Rewrite_Shop/buy.php?id=-1%20OR%2017-7%3d10
- http://testphp.vulnweb.com/secured/newuser.php
- http://testphp.vulnweb.com/userinfo.php

Description

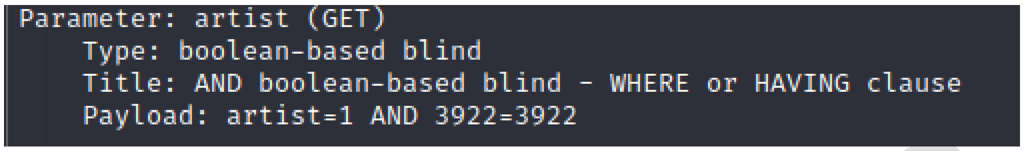
I confirmed it is has SQL Query by executing different Advanced SQL Map tests on the backend database. In these tests, SQL injection was not obvious, but the different responses from the page based on the injection test allowed me to identify and confirm the SQL injection presence.

e.g



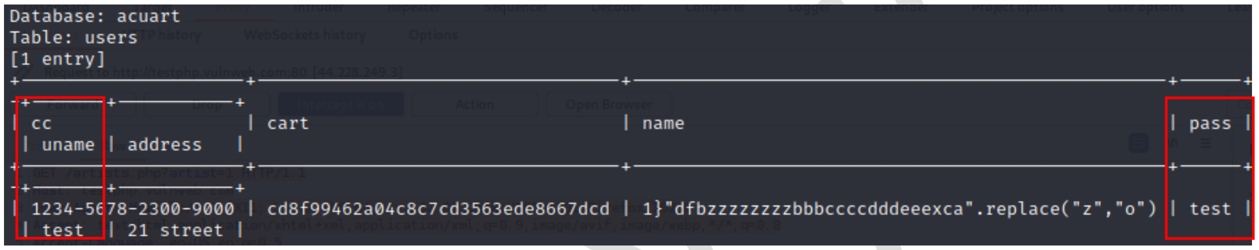
```
[08:23:11] [INFO] GET parameter 'artist' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --string="sem")
```

and by checking via the databases using [sqlmap -r sqlmap.txt --dbs]



```
Parameter: artist (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: artist=1 AND 3922=3922
```

With this command [sqlmap -r sqlmap.txt --dbs --tables -D acuart -T users --dump]I got here



```
Database: acuart
Table: users
[1 entry]
+---------------------+----------------------------------+-----------------------------------------------------+------+
+-------+-------------+
| cc                  | cart                             | name                                                | pass |
| uname | address     |
+---------------------+----------------------------------+-----------------------------------------------------+------+
+-------+-------------+
| 1234-5678-2300-9000 | cd8f99462a04c8c7cd3563ede8667dcd | 1}"dfbzzzzzzzzbbbccccdddeeexca".replace("z","o") | test |
| test  | 21 street |
```

Depending on the backend database, the database connection settings and the operating system, an attacker can mount one or more of the following type of attacks successfully:

- Reading, updating and deleting arbitrary data/tables from the database
- Executing commands on the underlying operating system

Solution

The best way to protect your code against SQL injections is using parameterized queries (*prepared statements*). Almost all modern languages provide built-in libraries for this. Wherever possible, do not create dynamic SQL queries or SQL queries with string concatenation.

- [SQL injection Prevention Cheat Sheet]
- [A guide to preventing SQL injection]

Cross-Site Scripting (XSS)

(Hijacking user's active session, Mounting phishing attacks, Intercepting data and performing man-in-the-middle attacks)

Target: http://testphp.vulnweb.com

Technical severity: **high**

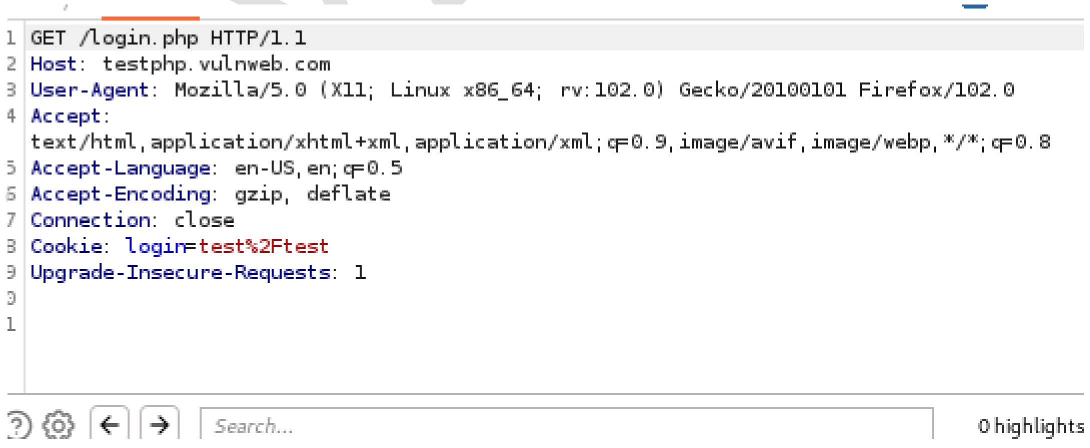
Vulnerability details:

- http://testphp.vulnweb.com/comment.php
- http://testphp.vulnweb.com/guestbook.php
- http://testphp.vulnweb.com/hpp/params.php?aaaa%2f=&p=valid&pp=%3cscRipt%3enetsparker(0x01B4B3)%3c%2fscRipt%3e
- http://testphp.vulnweb.com/search.php?test=query
- http://testphp.vulnweb.com/secured/newuser.php

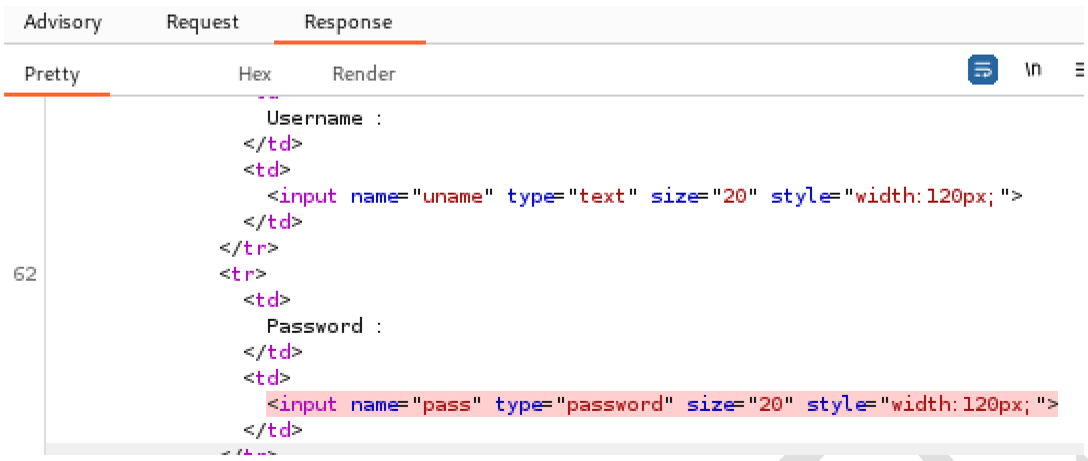
Description

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 a 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 a limitation, since it allows attackers to hijack other users' sessions, an attacker might attack an administrator to gain full control over the application.

The page contains a form with the above action URL and the form contains the following password field with autocomplete enabled: pass



```
GET /login.php HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: login=test%2Ftest
Upgrade-Insecure-Requests: 1
```



```
          Username :
        </td>
        <td>
          <input name="uname" type="text" size="20" style="width:120px;">
        </td>
      </tr>
      <tr>
        <td>
          Password :
        </td>
        <td>
          <input name="pass" type="password" size="20" style="width:120px;">
        </td>
```

Solution

To prevent browsers from storing credentials entered into HTML forms, include the attribute autocomplete="off" within the FORM tag (to protect all form fields) or within the relevant INPUT tags (to protect specific individual fields).

Please note that modern web browsers may ignore this directive. In spite of this there is a chance that not disabling autocomplete may cause problems obtaining PCI compliance.

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.

- CWE-200: Information Exposure
- OWASP XSS Prevention Cheat Sheet
- OWASP AntiSamy Java

Tools

- Zofixer.com
- Google dork
- Nmap
- Burp-suite Professional
- Whatweb
- SQL Map Test
- Whois